

ĐẠI HỌC BÁCH KHOA HÀ NỘI  
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

\*



# PROJECT 1

## MẬT MÃ HỌC (Cryptography)

Họ và tên	: Phan Thế Toàn
Mã số sinh viên	: 20225415
Lớp	: Kỹ thuật máy tính 02-K67
Giảng viên hướng dẫn	: TS. Trần Vĩnh Đức : TS. Đỗ Tiến Dũng

Hà Nội, tháng 12 năm 2024

# MỤC LỤC

<b>MỤC LỤC .....</b>	<b>2</b>
<b>DANH MỤC HÌNH ẢNH .....</b>	<b>4</b>
<b>CHƯƠNG 1. GIỚI THIỆU CHUNG .....</b>	<b>6</b>
1.1. Định nghĩa Mật mã học.....	6
1.2. Các khái niệm trong hệ thống mật mã .....	6
1.3. Ý nghĩa và Ứng dụng.....	7
1.3.1. Ý nghĩa .....	7
1.3.2. Ứng dụng.....	8
<b>CHƯƠNG 2. MẬT MÃ CỔ ĐIỂN.....</b>	<b>9</b>
<b>CHƯƠNG 3. MẬT MÃ KHÓA DÒNG .....</b>	<b>11</b>
3.1. One Time Pad.....	11
3.2. PRGs .....	12
3.3. Semantic Security .....	14
3.4. Mã dòng Rivest Cipher 4 .....	15
<b>CHƯƠNG 4. MẬT MÃ KHỐI.....</b>	<b>18</b>
4.1. PRFs & PRPs .....	19
4.2. Thuật toán DES.....	20
4.3. Thuật toán 3DES.....	28
4.4. Thuật toán AES.....	29
4.5. One Time Key .....	39
4.5.1. ECB .....	39
4.6. Many Time Key.....	41
4.6.1. CPA .....	41
4.7. Các chế độ mã khống .....	43
4.7.1. CBC .....	43
4.7.2. CTR .....	44
<b>CHƯƠNG 5. TOÀN VẸN THÔNG ĐIỆP .....</b>	<b>46</b>
5.1.1. MAC.....	46
5.1.2. MAC dựa trên PRF.....	47
5.1.3. CBC-MAC & NMAC .....	47
<b>CHƯƠNG 6. HÀM BĂM MẬT MÃ .....</b>	<b>50</b>
6.1.1. Thuật toán hàm băm SHA-256A.....	50

## **IT3150 – Project 1**

---

<b>CHƯƠNG 7. MẬT MÃ XÁC THỰC .....</b>	<b>55</b>
7.1. Tán công CPA bảo mật .....	55
7.2. Định nghĩa Mã xác thực .....	56
7.3. CCA.....	56
7.3.1. Xây dựng từ hệ mã và MACs.....	58
7.3.2. OCB.....	59
<b>CHƯƠNG 8. CÁC GIAO THỨC MẬT MÃ HỌC .....</b>	<b>60</b>
8.1. Giao thức Diffie-Hellman.....	60
<b>CHƯƠNG 9. MẬT MÃ KHÓA CÔNG KHAI.....</b>	<b>63</b>
9.1. TDF .....	63
9.2. RSA .....	64
9.3. RSA-TDP .....	64
9.4. One-way function.....	64
9.5. Hệ mật Elgamal ( <i>hiện đại</i> ) .....	65
<b>CHƯƠNG 10. KẾT LUẬN.....</b>	<b>67</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>68</b>

## **DANH MỤC HÌNH ẢNH**

Hình 1. Biểu đồ phân bố các chữ cái trong một mẫu văn bản tiếng Anh.....	10
Hình 2. Phép toán XOR.....	11
Hình 3. Bộ sinh số giả ngẫu nhiên.....	13
Hình 4. Mô tả Semantic Security .....	14
Hình 5. OTP is semantically secure .....	15
Hình 6. RC4 block diagram.....	15
Hình 7. Quá trình tạo mã RC4.....	16
Hình 8. Tóm tắt quá trình tạo mã .....	17
Hình 9. Sơ đồ tổng quát mô hình mã khối.....	18
Hình 10. Sơ đồ cơ bản DES.....	20
Hình 11. Ý tưởng mạng Feistel Network .....	20
Hình 12. Sơ đồ khối của DES.....	21
Hình 13. Cấu trúc thuật toán DES – Tạo mã .....	21
Hình 14. Initial permutation .....	22
Hình 15. Hàm vòng – Round Function .....	22
Hình 16. Mô tả hàm F .....	23
Hình 17. Expansion E .....	24
Hình 18. S-box.....	24
Hình 19. Ví dụ S-box 5.....	24
Hình 20. Bộ đầy đủ 8 S-box theo DES.....	25
Hình 21. Bảng hoán vị P .....	25
Hình 22. Hàm sinh khóa .....	26
Hình 23. Input key .....	26
Hình 24. Permuted Choice One (PC-1) .....	27
Hình 25. Permuted Choice Two (PC-2) .....	27
Hình 26. Schedule of Left Shifts .....	27
Hình 27. Thay đổi bản rõ và khóa .....	28
Hình 28. Kết quả các giá trị trung gian thuật toán .....	28
Hình 29. Sơ đồ hoạt động 3DES .....	29
Hình 30. AES là mạng thay thế hoán vị (Subs-Perm network).....	30
Hình 31. AES-128 .....	30
Hình 32. The round function .....	31
Hình 33. Thuật toán Mã hóa – Giải mã AES .....	31
Hình 34. Mô tả chi tiết thuật toán AES .....	32
Hình 35. Substitute Bytes Transformation .....	32
Hình 36. Shift row transformation.....	33
Hình 37. Mix column transformation.....	33
Hình 38. Add round key transformation.....	34
Hình 39. Key function generator – AES key expansion .....	34
Hình 40. Key expansion for AES example .....	37
Hình 41. AES example .....	38
Hình 42. Quá trình thay đổi bản rõ .....	39

## **IT3150 – Project 1**

---

Hình 43. Hoạt động ECB.....	39
Hình 44. Mã hóa ảnh bằng ECB.....	40
Hình 45. ECB is not Semantically Secure.....	41
Hình 46. Hoạt động của CPA trong khóa nhiều lần.....	41
Hình 47. Bản mã không bảo mật dưới tấn công CPA .....	42
Hình 48. Hàm mã hóa ngẫu nhiên .....	42
Hình 49. Sơ đồ mã hóa theo chế độ CBC.....	43
Hình 50. Sơ đồ giải mã CBC .....	44
Hình 51. CBC - Padding.....	44
Hình 52. Sơ đồ mã hóa CTR .....	45
Hình 53. So sánh CBC & CTR mode.....	45
Hình 54. Message integrity: MACs.....	46
Hình 55. MACs – Secret Key.....	46
Hình 56. MACs – PRF .....	47
Hình 57. CBC-MAC.....	48
Hình 58. NMAC .....	48
Hình 59. MAC Padding.....	49
Hình 60. Hoạt động của hàm băm .....	50
Hình 61. One iteration in a SHA-2 family compression function.....	51
Hình 62. Toàn vẹn Ciphertext .....	56
Hình 63. Kết hợp MAC và ENC (CCA) .....	58
Hình 64. OCB .....	59
Hình 65. Thuật toán trao đổi khóa Diffie-Hellman .....	61
Hình 67. Hiệu năng Elgamal .....	65
Hình 68. Elgamal – CCA.....	66

# CHƯƠNG 1. GIỚI THIỆU CHUNG

## 1.1. Định nghĩa Mật mã học

Mật mã học (Cryptography) là lĩnh vực nghiên cứu và thực hành liên quan đến việc bảo vệ thông tin bằng cách sử dụng các kỹ thuật biến đổi dữ liệu thành dạng không thể đọc được cho những người không có quyền truy cập. Mục tiêu chính của mật mã học là đảm bảo tính bảo mật, xác thực và toàn vẹn của thông tin.

Mật mã là một công cụ mạnh mẽ đồng thời là nền tảng của nhiều cơ chế bảo mật và mật mã không phải là giải pháp của mọi vấn đề và không đáng tin cậy nếu không được triển khai và sử dụng đúng cách.

## 1.2. Các khái niệm trong hệ thống mật mã

- Một hệ thống mật mã là một bộ năm

$$S = (P, C, K, E, D)$$

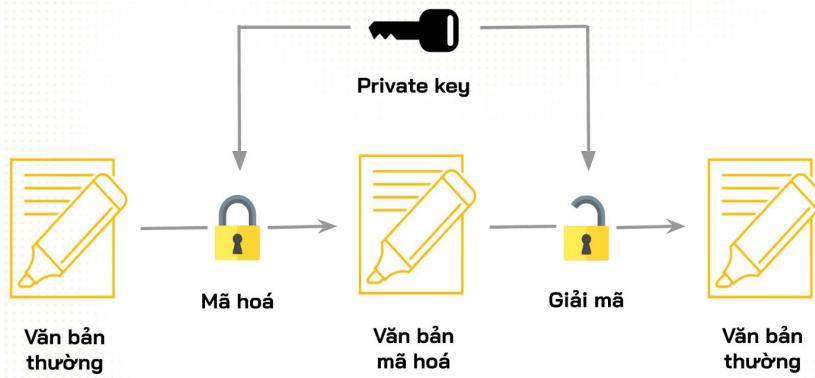
Thỏa mãn các tập điều kiện sau đây:

- Tập nguồn **P (plaintext)** là tập hữu hạn tất cả bản tin nguồn cần mã hóa có thể có
- **C (ciphertext)** là một tập hữu hạn các ký tự bản mã
- **K (key)** là tập hữu hạn các khóa có thể được sử dụng
- **E (encryption)** là một ánh xạ từ  $(K, P)$  vào  $C$ , được gọi là phép lập mật mã
- **D (decryption)** là một ánh xạ từ  $(K, C)$  vào  $P$ , được gọi là phép giải mã

*Chú ý: các ánh xạ này là ánh xạ 1-1*

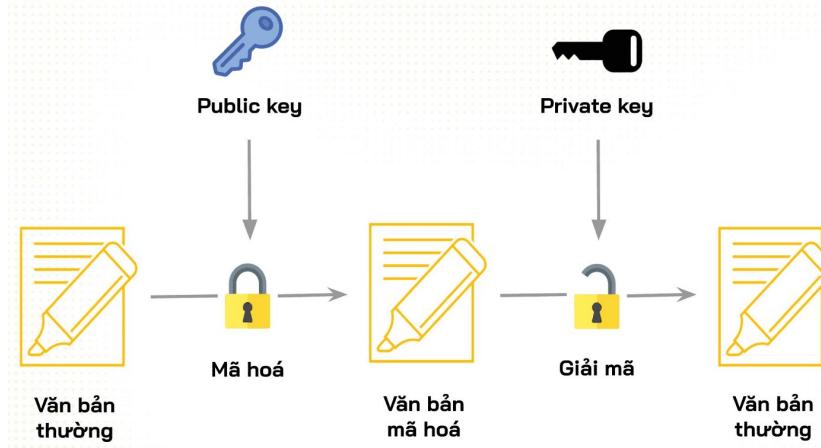
- Mã hóa đối xứng (*symmetric encryption*) là một phương pháp mã hóa trong đó cùng một khóa **k** được sử dụng để mã hóa và giải mã dữ liệu. Điều này có nghĩa là cả người gửi và người nhận đều phải giữ bí mật khóa này để bảo vệ thông tin.

## SYMMETRIC ENCRYPTION



- Mã hóa bắt đôi xứng (*asymmetric encryption*) là phương pháp mã hóa sử dụng hai khóa riêng biệt: khóa công khai (*public key*) và khóa riêng tư (*private key*) để bảo mật thông tin. Khóa công khai có thể được chia sẻ rộng rãi, trong khi khóa riêng tư phải được giữ bí mật.

## ASYMMETRIC ENCRYPTION



### 1.3. Ý nghĩa và Ứng dụng

#### 1.3.1. Ý nghĩa

Mật mã học giúp bảo vệ dữ liệu khỏi việc bị truy cập trái phép, đảm bảo rằng chỉ những người có quyền mới có thể đọc hoặc chỉnh sửa thông tin. Mật mã xác nhận danh tính của người gửi và người nhận đồng thời ngăn chặn việc giả mạo thông tin, đảm bảo thông tin không bị thay đổi trong quá trình truyền tải, giúp người dùng yên tâm về

độ chính xác của dữ liệu. Tạo ra môi trường giao tiếp an toàn, từ đó tăng cường sự tin tưởng các bên trong giao dịch

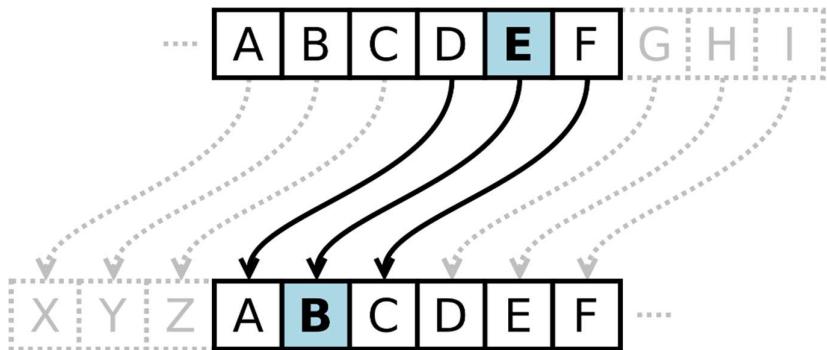
### 1.3.2. Ứng dụng

Mật mã học được áp dụng rộng rãi trong nhiều lĩnh vực khác nhau, đóng vai trò quan trọng trong việc bảo vệ thông tin và đảm bảo an toàn trong giao tiếp. Một trong những ứng dụng nổi bật:

- Giao tiếp an toàn: Sử dụng trong các ứng dụng như email, tin nhắn và giao dịch trực tuyến để bảo vệ thông tin
- Chữ ký số: Được sử dụng để xác thực tài liệu điện tử, đảm bảo rằng tài liệu không bị thay đổi và người ký là ai.
- Bảo mật dữ liệu: Mật mã học được áp dụng để bảo vệ dữ liệu trên đĩa cứng, thiết bị di động và dịch vụ lưu trữ đám mây.
- Giao thức bảo mật: Sử dụng trong các giao thức như SSL/TLS để bảo vệ thông tin truyền tải qua Internet.
- Tiền điện tử: Mật mã học là nền tảng cho các loại tiền điện tử như Bitcoin, giúp đảm bảo tính bảo mật và ẩn danh trong giao dịch.

## CHƯƠNG 2. MẬT MÃ CỔ ĐIỂN

Mật mã Caesar là một trong những kỹ thuật mã hóa đơn giản và phổ biến nhất. Đây là một dạng mật mã thay thế, trong đó mỗi ký tự trên bản rõ sẽ được thay bằng một ký tự khác, có vị trí cách nó một khoảng xác định trong bảng chữ cái. Ví dụ với độ dịch chuyển là 3, D sẽ trở thành A, E sẽ trở thành B,...



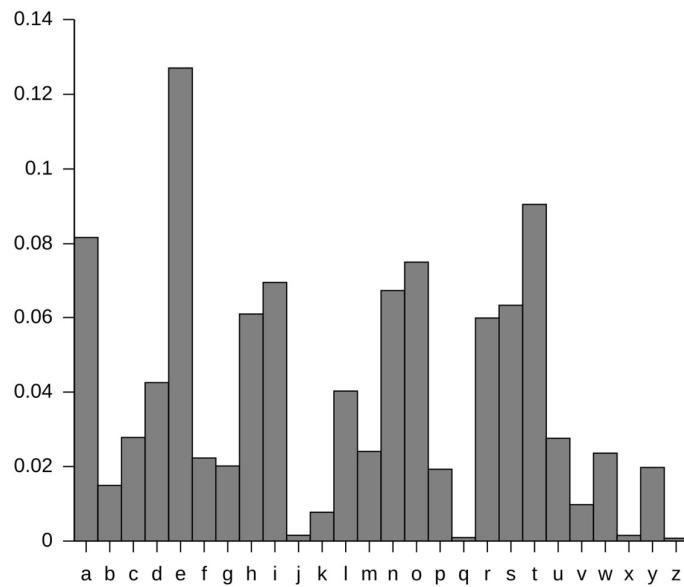
Ví dụ:

Bản rõ	THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Bản mã	WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ

Ngoài ra còn có thể mã hóa cũng có thể được biểu diễn thông qua số học mô đun, bằng cách gán các ký tự bằng các con số, theo tuần tự,  $A \rightarrow 0, B \rightarrow 1, \dots, Z \rightarrow 25$ . Mã hóa một chữ cái  $x$  bằng phép dịch chuyển  $n$  vị trí có thể mô tả bằng biểu thức toán học sau:

- Mã hóa:  $E_n(x) = (x+n) \bmod 26$
- Giải mã:  $D_n(x) = (x-n) \bmod 26$

Độ bảo mật của mật mã Caesar rất dễ bị phá vỡ. Với chỉ 25 khả năng dịch chuyển (1 đến 25), kẻ tấn công có thể dễ dàng thử tất cả các khả năng để tìm ra văn bản gốc chẳng hạn như phân tích tần suất hay phân tích các từ mẫu hoặc có thể sử dụng tấn công vét cạn để kiểm tra lần lượt.



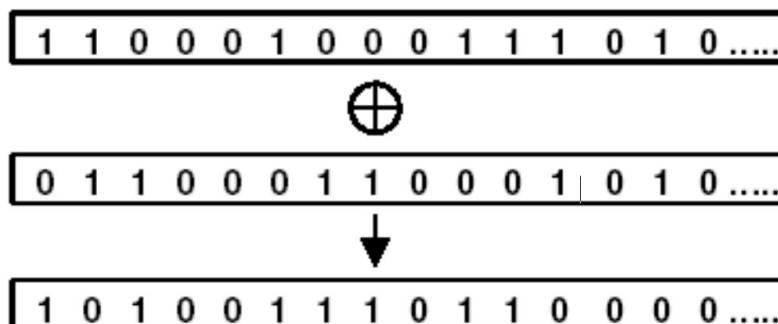
*Hình 1. Biểu đồ phân bố các chữ cái trong một mẫu văn bản tiếng Anh  
Phép dịch chuyển Caesar "xoay" biểu đồ này và có thể xác định bằng cách xem xét  
biểu đồ*

## CHƯƠNG 3. MẬT MÃ KHÓA DÒNG

### 3.1. One Time Pad

One-Time Pad (OTP) là một phương pháp mã hóa tuyệt đối an toàn, được phát triển vào đầu thế kỷ 20. Nó sử dụng một chuỗi ngẫu nhiên (**key**) có độ dài bằng hoặc lớn hơn thông điệp cần mã hóa.

Nguyên tắc hoạt động sử dụng chuỗi key phải hoàn toàn ngẫu nhiên, không được lặp lại và chỉ sử dụng một lần duy nhất. Trong quá trình mã hóa mỗi ký tự của thông điệp được mã hóa bằng cách thực hiện phép **XOR** (hoặc cộng modulo) với ký tự tương ứng trong chuỗi key. Để giải mã thì thực hiện ngược lại tương tự với chuỗi key tương ứng



Hình 2. Phép toán XOR

➤ Mã hóa:  $c_i = (m_i \oplus k_i)$

### One-time Pad: Encryption

e=000 h=001 i=010 k=011 l=100 r=101 s=110 t=111
---

Encryption: Plaintext  $\oplus$  Key = Ciphertext

Plaintext:	h	e	i	l	h	i	t	l	e	r
Key:	001	000	010	100	001	010	111	100	000	101
	111	101	110	101	111	100	000	101	110	000
Ciphertext:	110	101	100	001	110	110	111	001	110	101

s r l h s s t h s r

- Giải mã:  $m_i = (m_i \oplus k_i) \oplus k_i = m_i \oplus (k_i \oplus k_i)$

## One-Time Pad: Decryption

e=000 h=001 i=010 k=011 l=100 r=101 s=110 t=111

**Decryption:** Ciphertext  $\oplus$  Key = Plaintext

	s	r	l	h	s	s	t	h	s	r
Ciphertext:	110	101	100	001	110	110	111	001	110	101
Key:	111	101	110	101	111	100	000	101	110	000
Plaintext:	001	000	010	100	001	010	111	100	000	101

h e i l h i t l e r

Với khóa  $k$  thì phải đáp ứng 3 điều kiện sau đây để đảm bảo tính bảo mật

- Độ dài của khóa phải bằng kích thước bản rõ.
- Khóa phải được chọn hoàn toàn ngẫu nhiên (truly random).
- Khóa chỉ được sử dụng một lần.

$$\Pr[E(k, m) = c] = \frac{\#\{k \in \mathcal{K} \text{ thỏa mãn } E(k, m) = c\}}{\#\mathcal{K}}$$

với  $\#\{k \in K \text{ thỏa mãn } E(k, m) = c\} = const$

- Nếu thỏa mãn 3 điều kiện trên, hệ mã **OTP** sẽ là an toàn tuyệt đối (*perfect security*) theo định lý của Clause Shannon.

Tuy nhiên phương pháp One-Time Pad không có ý nghĩa sử dụng thực tế. Vì chiều dài khóa bằng chiều dài bản tin, mỗi khóa chỉ sử dụng một lần, nên thay vì truyền khóa trên kênh an toàn thì có thể truyền trực tiếp bản rõ mà không cần quan tâm đến vấn đề mã hóa

## 3.2. PRGs

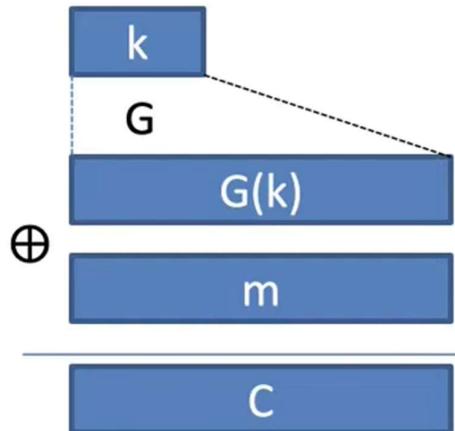
Bộ sinh số giả ngẫu nhiên (Pseudorandom Generators - PRGs) là các thuật toán hoặc hàm có khả năng tạo ra một chuỗi số mà có tính chất giống như số ngẫu nhiên, nhưng thực tế là được sinh ra từ một chuỗi đầu vào (seed) có độ dài ngắn hơn.

$$G: \{0, 1\}^s \rightarrow \{0, 1\}^n \text{ với } n \gg s.$$

Các phép toán thông qua thuật toán xác định (*deterministic algorithm*)

$$C = E(k, m) = m \oplus G(k)$$

$$D(k, C) = C \oplus G(k)$$



Hình 3. Bộ sinh số giả ngẫu nhiên

Trong đó  $G(k)$  là các thuật toán xác định, ví dụ như thanh ghi dịch tuyến tính (LFSR), random.org, blum blum slub,... Tính chất yêu cầu của PRGs để mã dòng được bảo mật đó là chuỗi số được sinh ra phải có tính chất ngẫu nhiên, nghĩa là không thể dự đoán được từ chuỗi đầu vào, xác suất phân phối các số sinh ra phải được phân phối đều và không lặp lại, hiệu suất PRGs phải chạy nhanh và hiệu quả để có thể sinh ra một lượng lớn số ngẫu nhiên từ một chuỗi đầu vào nhỏ (*seed s*)

Giả sử PRGs có thể dự đoán được  $\exists i: G(k)|_{1,2,\dots,i} \rightarrow G(k)|_{1,2,\dots,n}$  do đó mặc dù đây là một vấn đề lớn, tuy nhiên PRGs sẽ có thể dự đoán được nếu như  $G: k \rightarrow \{0,1\}^n$

$$\exists \text{'eff' alg. } A \text{ và } \exists a \leq i \leq n-1$$

(A là thuật toán hiệu quả nào đó)

Khi đó:  $Pr_{k \leftarrow K}[A(G(k))|_{1,2,\dots,i} = G(k)|_{i+1}] > \frac{1}{2} + \varepsilon$  với ( $\varepsilon \sim \frac{1}{2}^{30}$ ).

Tức là khi biết được  $G(k)|_{1,2,\dots,i}$  thì có thể dự đoán được  $G(k)|_{i+1}$  nếu tồn tại một thuật toán A và xác suất của thuật toán có thể dự đoán lớn hơn  $\frac{1}{2} + \varepsilon$  (*lợi thế khả năng dự báo*). Vậy PRG sẽ không thể dự đoán được nếu nó không có khả năng dự đoán khi  $\forall i$  thì không có một lợi thế đáng kể nào có thể dự đoán được ( $i+1$ ) tương đương với “non-neg”  $\varepsilon$ .

#### ➤ Bảo mật PRGs

Với  $G: k \rightarrow \{0,1\}^n$  là một PRG và yêu cầu PRG phải không nhận diện được, đầu ra là chuỗi thật sự ngẫu nhiên, nhận giá trị 0, 1 ngẫu nhiên:  $P(A(r)) = 1 \parallel 0 = \frac{1}{2}$  Thuật

toán tác động lên kiểm định thống kê  $A(x)$  trên  $\{0,1\}^n$  với đầu ra “0” hoặc “1” với đầu ra 0 nghĩa là không ngẫu nhiên và 1 nghĩa là ngẫu nhiên

$$Adv_{PRG}[A, G] = | \Pr_{k \leftarrow K}[A(G(k)) = 1] - \Pr_{r \leftarrow \{0,1\}^n}[A(G(k)) = 1] | \in [0, 1]$$

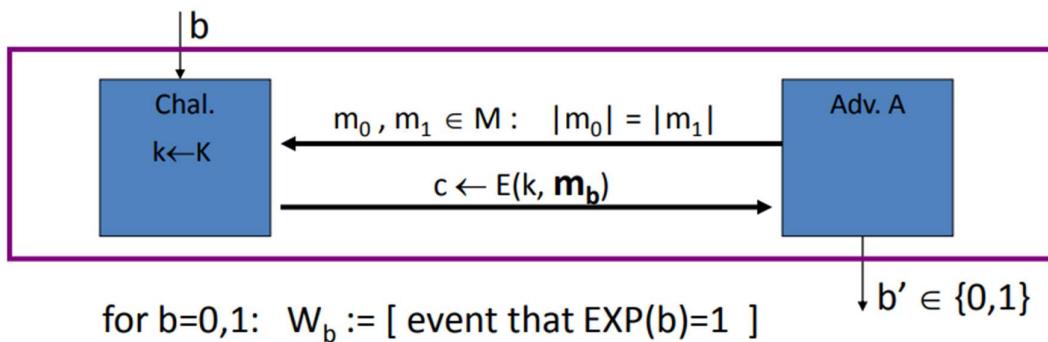
- $Adv \sim 1$ : alg. A có thể nhận diện được PRG sinh ra từ tập ngẫu nhiên
- $Adv \sim 0$ : alg. A không thể nhận diện được đầu vào là G

Vậy để có một PRG bảo mật nếu tồn tại thuật toán A với điều kiện xác xuất dự đoán nhỏ hơn một số không đáng kể và lợi thế nhận diện được sinh ra từ tập G hay từ tập thật sự ngẫu nhiên gần 0.

### 3.3. Semantic Security

Bảo mật ngữ nghĩa (Semantic Security) định nghĩa rằng nếu một hệ thống mã hóa được coi là an toàn nếu không có thông tin nào về thông điệp gốc có thể được suy ra từ bản mã, ngay cả khi kẻ tấn công có quyền truy cập vào bản mã đó.

For  $b=0,1$  define experiments EXP(0) and EXP(1) as:



$$Adv_{SS}[A, E] := | \Pr[W_0] - \Pr[W_1] | \in [0,1]$$

Hình 4. Mô tả Semantic Security

Giả sử kẻ tấn công  $A$  xâm nhập vào hệ thống, có thể tự tạo ra các bản rõ  $m_0, m_1$  và thu các bản mã  $c$  từ mã hóa tuy nhiên không biết về khóa  $k$ . Dựa trên hai bản rõ  $m_0, m_1$  có độ dài như nhau được gửi đi đồng thời và nhận được bản mã  $c$  tuy nhiên để nhận biết được  $c$  được mã hóa từ  $m_0$  hay  $m_1$  chưa nhận biết được. Nếu kẻ tấn công không tìm ra được một kiểm định thống kê nào đó để suy ra

$$c_0 = [E(k, m_0) \parallel E(k, m_1)]$$

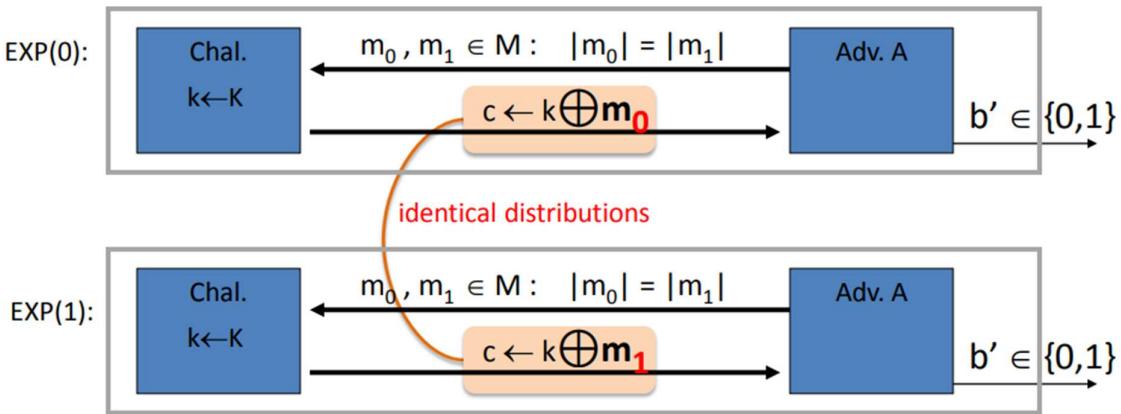
được tạo ra từ  $m_0$  hay  $m_1$  thì hệ thống của bạn sẽ đảm bảo tính chất an toàn bảo mật về ngữ nghĩa

$$Adv_{ss} \sim 0 : \text{không thể phân biệt}$$

$$Adv_{ss} \sim 1 : \text{có thể phân biệt}$$

Tuy nhiên nếu tồn tại kiểm định thống kê để phân biệt thì kẻ tấn công có thể phân biệt được bản mã tạo ra từ  $m_0$  hoặc  $m_1$  và từ đó tính ra được khóa  $k$

Ví dụ: One Time Pad (*OTP*) là một bảo mật ngữ nghĩa



Hình 5. *OTP is semantically secure*

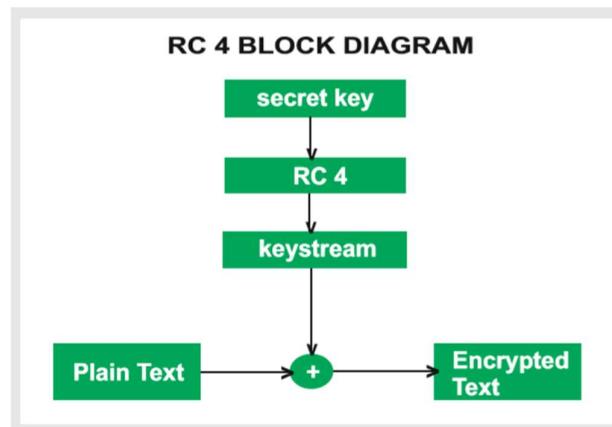
Với khóa một lần khi kẻ tấn công gửi đi 2 bản tin có độ dài giống nhau và mỗi lần mã hóa là một khóa riêng biệt từ đó dẫn tới xác xuất để nhận biết đâu là của bản tin  $m_1$  hoặc  $m_0$  là giống nhau và lợi thế tấn công lúc này

$$Adv_{ss}[A, E] = | Pr[A(k \oplus m_0)] - Pr[A(k \oplus m_1)] | = 0$$

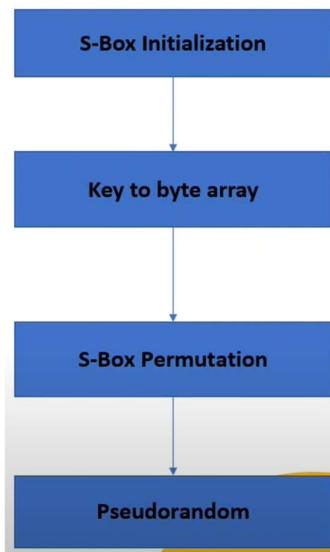
Với lợi thế bằng 0 thì kẻ tấn công sẽ không nhận diện được. Hệ thống bảo mật an toàn về ngữ nghĩa

### 3.4. Mã dòng Rivest Cipher 4

Rivest Cipher 4 là một hệ mật mã dòng, có độ dài khóa không cố định từ 1 – 256 bytes. Từ khóa bí mật ban đầu, tạo ra dòng khóa giả ngẫu nhiên rồi thực hiện phép XOR với bản tin rõ để tạo bản tin mật, ngược lại quá trình giải mã ta sẽ đem bản tin mật XOR với keystream sẽ thu được bản rõ

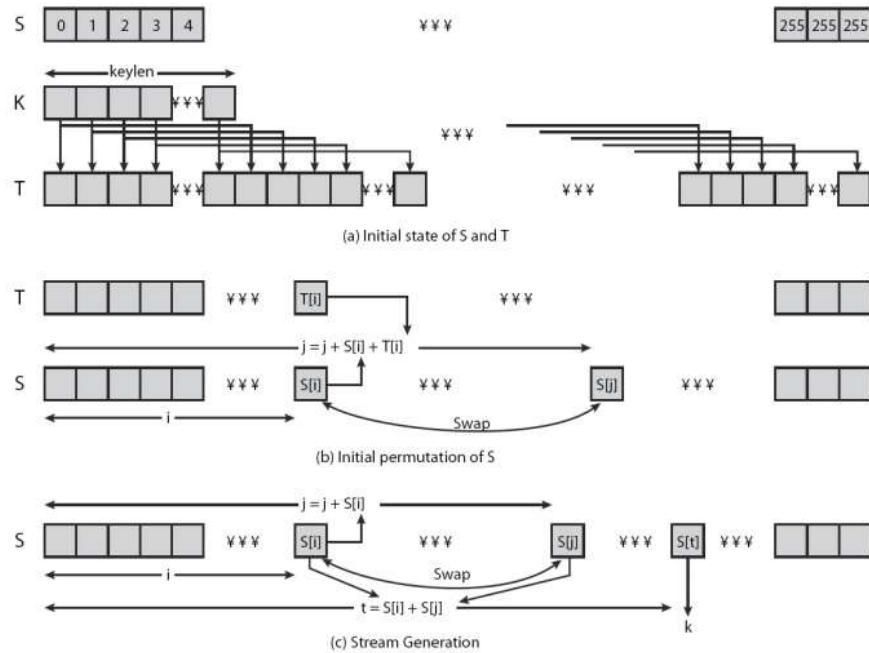


Hình 6. RC4 block diagram



Hình 7. Quá trình tạo mã RC4

- Giai đoạn khai tạo S  
S là một mảng số tăng dần từ 0 đến 255:  $S[0] = 0, S[1] = 1, \dots, S[255] = 255$ .  
Mảng tạm thời T cũng được tạo ra chính là khóa sao chép từ khóa bí mật ban đầu được lặp đi lặp lại để lắp đầy 256 bytes
- Hoàn vị ban đầu của S  
Thuật toán lập lịch khóa(KSA) được sử dụng để khai tạo hoán vị trong mảng “S”. Chúng ta có thể sử dụng T để tạo ra các hoán vị ban đầu của S. Ban đầu ta sẽ tạo  $S[i]: S[0] \rightarrow S[255]$ , ở bước này ta tạo thêm  $S[j]$  hoán vị theo lược đồ tạo ra bởi  $T[i]$
- Sinh số giả ngẫu nhiên PRGA  
Sau khi hoàn thành quá trình hoán vị ban đầu của S thì khóa ban đầu hoàn toàn không được sử dụng nữa. Mỗi K thu được đem XOR với ký tự tương ứng ở bản rõ để thu được ký tự ở bản mật



#### RC4 Overview

Hình 8. Tóm tắt quá trình tạo mã

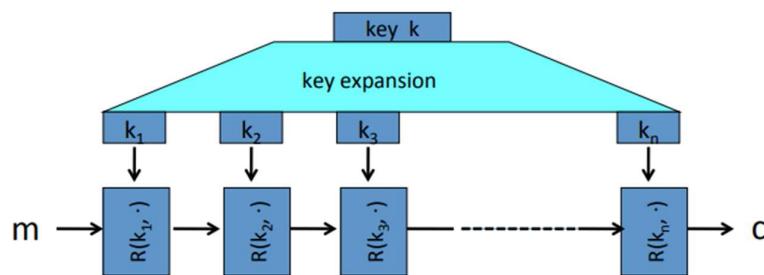
RC4 được ứng dụng trong HTTPs và WEP.

Tính bảo mật tổng thể của RC4 với khả năng phân tích các vấn đề với sự phân phối không đều và khả năng xuất hiện của các byte giống nhau tạo ra rủi ro cho bảo mật. Điều này có thể dẫn đến việc kẻ tấn công có thể dự đoán hoặc phân tích dữ liệu mã hóa. Do những lỗ hổng này, RC4 không còn được coi là an toàn cho các ứng dụng bảo mật hiện đại. Các thuật toán thay thế như AES được khuyến nghị sử dụng thay vì RC4. RC4 có một số điểm yếu nghiêm trọng trong tính bảo mật, bao gồm sự phân phối không đều của các byte và khả năng bị tấn công liên quan đến khóa. Do đó, nó không còn được khuyến nghị cho các ứng dụng bảo mật hiện đại

## CHƯƠNG 4. MẬT MÃ KHỐI

Mật mã khối (Block Cipher) là một phương pháp mã hóa trong đó dữ liệu được chia thành các khối có kích thước cố định và mã hóa từng khối một. Mỗi khối dữ liệu sẽ được xử lý độc lập bằng một thuật toán mã hóa và một khóa bí mật.

Cấu trúc của mã khối bao gồm các khối dữ liệu đầu vào được chia thành các khối, thường có kích thước 64 bit hoặc 128 bit, mỗi khối được mã hóa bằng một khóa bí mật, có thể có độ dài khác nhau (Ví dụ: 128, 192 hoặc 256 bit), về kích thước của các khóa, khối đầu vào là cố định và sử dụng thuật toán mã hóa để biến đổi đầu vào thành dữ liệu mã hóa



Hình 9. Sơ đồ tổng quát mô hình mã khối

- Key  $k$ : Khóa bí mật để mã hóa/giải mã dữ liệu.
- Key *expansion*: Quá trình mở rộng ban đầu thành các khóa con sử dụng trong các vòng lặp của thuật toán mã hóa.
- $k_1, k_2, k_3, \dots, k_n$ : Các khóa con được tạo ra từ quá trình mở rộng khóa.
- $m$ : Dữ liệu cần được mã hóa – plaintext
- $R(k_i, m)$ : Hàm mã hóa sử dụng khóa  $k$  và dữ liệu  $m$  để tạo ra dữ liệu mã hóa
- $c$ : Dữ liệu mã hóa (ciphertext) được tạo ra từ quá trình mã hóa

Mã khối thực hiện dựa trên kỹ thuật lặp, trong đó dữ liệu đầu vào được chia nhỏ ra thành các khối có kích thước bằng nhau và không đổi, mỗi khối sẽ được mã hóa độc lập cùng một khóa bí mật, khóa này sẽ tạo sinh ra  $n$  khóa tương ứng với  $n$  khối để mã hóa độc lập từng khối một thông qua hàm mã hóa  $R(k_i, m)$ . Mật mã khối có thể được sử dụng trong nhiều *chế độ* khác nhau để tăng cường tính bảo mật và tính linh hoạt

- ECB (Electronic Codebook): Mỗi khối được mã hóa độc lập.
- CBC (Cipher Block Chaining): Mỗi khối được mã hóa dựa trên khối trước đó, tạo ra sự phụ thuộc giữa các khối.
- CFB (Cipher Feedback): Mã hóa dữ liệu theo từng byte, cho phép mã hóa dữ liệu không phải là bội số của kích thước khối.
- OFB (Output Feedback): Tương tự như CFB nhưng không phụ thuộc vào dữ liệu đầu vào, tạo ra một chuỗi mã hóa độc lập.

- CTR (Counter Mode): Sử dụng một giá trị đếm để tạo ra các khối mã hóa, cho phép mã hóa song song và dễ dàng hơn trong việc xử lý.

Mật mã khối sử dụng các thuật toán mã hóa khác nhau, trong đó phổ biến AES (Advanced Encryption Standard) một trong những thuật toán mã hóa khối phổ biến và an toàn nhất hiện nay, DES (Data Encryption Standard) thuật toán cũ hơn, không còn được coi là an toàn do độ dài khóa ngắn, 3DES (Triple DES) cải tiến của DES bằng cách mã hóa ba lần với ba khóa khác nhau

## 4.1. PRFs & PRPs

Mật mã khối tạo ra các chuỗi số có độ dài xác định  $\{0,1\}^n \rightarrow \{0,1\}^n$ , không gian giá trị đầu vào và đầu ra là xác định và đây là một ánh xạ biến đổi từ tập X đến tập Y. Ánh xạ đó là một hàm giả ngẫu nhiên (*Pseudo Random Function - PRF*) ) được định nghĩa bởi  $(K, X, Y)$ . và một hoán vị giả ngẫu nhiên (*Pseudo Random Permutation - PRP*) được định nghĩa bởi  $(K, X)$ . Hàm giả ngẫu nhiên có đặc điểm là tập đầu vào X sẽ khác biệt với tập ra Y và được xác định thông qua một hàm (*function*) nào đó và tham số K

$$PRFs: K \times X \rightarrow Y$$

Do đó tồn tại một thuật toán hiệu quả để có thể ước lượng đầu ra của hàm  $F(k,x)$ . Hoán vị giả ngẫu nhiên có đặc điểm ánh xạ 1 điểm  $C_1$  thông qua một hoán vị giả ngẫu nhiên và tham số đầu vào K tạo ra được một  $C_2$  nhưng  $C_1, C_2$  đều thuộc tập X

$$PRPs: K \times X \rightarrow X$$

và sẽ tồn tại một thuật toán hiệu quả để ước lượng đầu ra của hàm  $E(k,x)$  ( $E(..)$  vẫn thuộc tập X do vậy hàm  $E(k,\bullet)$  là một song ánh) và cũng tồn tại một thuật toán nghịch đảo hiệu quả  $D(k,y)$

Ví dụ hoán vị giả ngẫu nhiên PRP: 3DES, AES

$$AES: K \times X \rightarrow X \text{ khi } K = X = \{0,1\}^{128}$$

$$3DES: K \times X \rightarrow X \text{ khi } K = \{0,1\}^{168}, X = \{0,1\}^{64}$$

và về nguyên lý để bất kỳ một PRP là một PRF khi  $X=Y$  và ngược lại. Tuy nhiên để PRF là một PRP khi PRF là một hàm song ánh và tồn tại một thuật toán hiệu D(k,y)

Để xét một hàm giả ngẫu nhiên bảo mật khi kết quả đầu ra được tạo từ tập ngẫu nhiên thật sự  $[X,Y]$  không thể phân biệt được với kết quả đầu ra từ một hàm giả ngẫu nhiên được tạo.

- Cho  $F: K \times X \rightarrow Y$  là một PRF:
- $Funcs[X,Y]$ : Tập tất cả các biến đổi  $X \rightarrow Y$
- $S_F = \{ F(k, \bullet), k \in K \} \subset Funcs[X,Y]$

Trong hàm  $Funcs[X,Y]$  là tổng hợp tất cả các hàm tạo  $X \rightarrow Y$  nhưng chưa phụ thuộc vào khóa K nhưng cho đầu ra  $Y_{k|1,2,\dots}$  với xác xuất phân bố đều (*gần ngẫu nhiên*),  $S_F$  là hàm sinh Y có phụ thuộc vào khóa K cho đầu ra  $Y_{k'|1,2,\dots}$ . Khi đó các *tính chất thống kê* của Y' sẽ không thể phân biệt được với các tính chất thống kê của Y nghĩa là không thể phân biệt được Y hay Y' được tạo ra từ  $S_F$  hay  $Funcs[X,Y]$  thì  $S_F$  sẽ là một PRF bảo mật.

Tương tự như vậy đối với hoán vị giả ngẫu nhiên bảo mật khi không thể phân biệt đầu ra là của một hoán vị ngẫu nhiên hay một hoán vị tập các hàm hoán vị *Perms[X]*

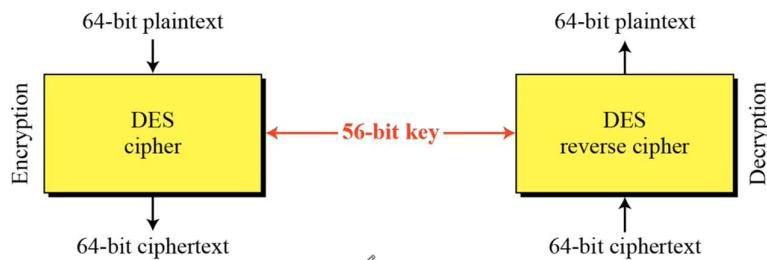
- Cho  $E: K \times X \rightarrow Y$  là một PRP:
- $\text{Perms}[X]$ : Tập các song ánh từ  $X \rightarrow Y$
- $S_E = \{E(k, \cdot), k \in K\} \subset \text{Perms}[X, Y]$

Giống với PRFs với tập các song ánh  $\text{Perms}[X]$  cho đầu ra  $Y$  và  $S_E$  cho đầu ra  $Y'$ . Hệ thống bảo mật khi đầu ra  $Y$  hoặc  $Y'$  không thể nhận biết được sinh từ  $\text{Perms}[X]$  hay  $S_E$  thì  $S_E$  sẽ là một PRP bảo mật

## 4.2. Thuật toán DES

- Giới thiệu chung về DES:

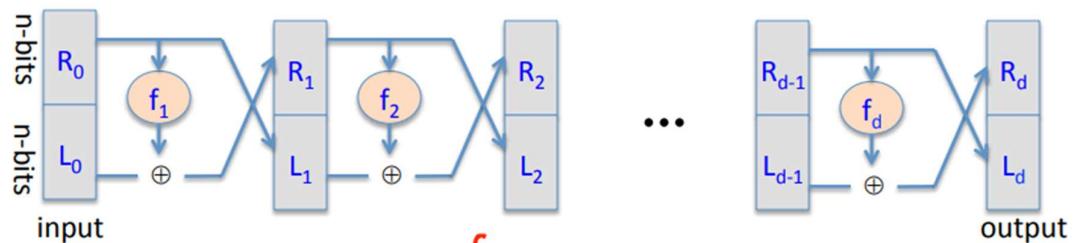
Data Encryption Standard (DES) được chấp nhận vào năm 1977 là tiêu chuẩn của Cục tiêu chuẩn quốc gia (National Bureau of Standards) và được Viện tiêu chuẩn và công nghệ quốc gia Hoa Kỳ (National Institute of Standards and Technology - NIST) chấp nhận là tiêu chuẩn liên bang (Federal Information Processing Standard 46 - FIPS PUB 46).



Hình 10. Sơ đồ cơ bản DES

Đặc điểm của thuật toán DES, đây là thuật toán mã khối, thuật toán sẽ chia input dữ liệu đầu vào thành các khối có độ dài là 64 bit, nếu khối cuối cùng không đủ 64 bit thì có thể *padding*,.... Khóa đầu vào dùng trong DES có độ dài toàn bộ là 64 bit. Tuy nhiên chỉ có 56 bit được thực sự sử dụng làm bit khóa còn 8 bit còn lại chỉ dùng cho việc kiểm tra(*checksum*). Khóa này sử dụng chung cho quá trình mã hóa và giải mã. Thuật toán được thực hiện qua 16 vòng mã hóa và được thiết kế để chạy trên hệ thống.

$$f_1, \dots, f_{16}: \{0,1\}^{32} \rightarrow \{0,1\}^{32} \quad f_i(x) = F(k_i, x)$$

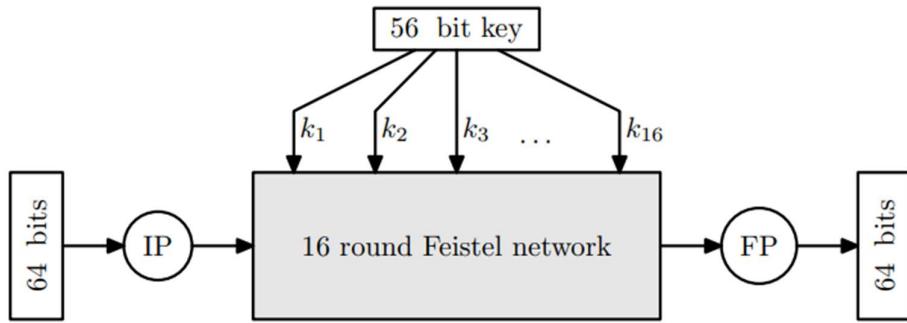


Hình 11. Ý tưởng mạng Feistel Network

Có:

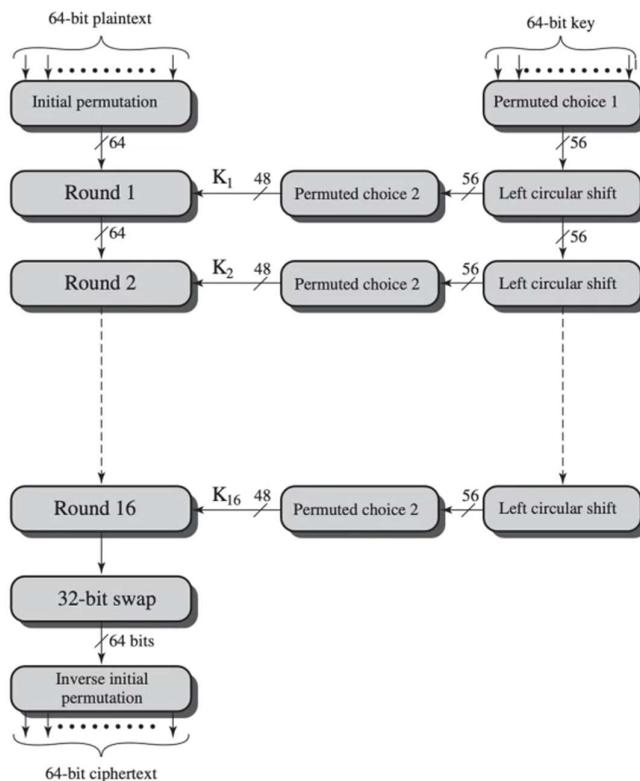
$$R_i = (F_i * R_{i-1}) \oplus L_{i-1}$$

$$L_i = R_{i-1}$$



Hình 12. Sơ đồ khối của DES

➤ Mô tả thuật toán



Hình 13. Cấu trúc thuật toán DES – Tạo mã

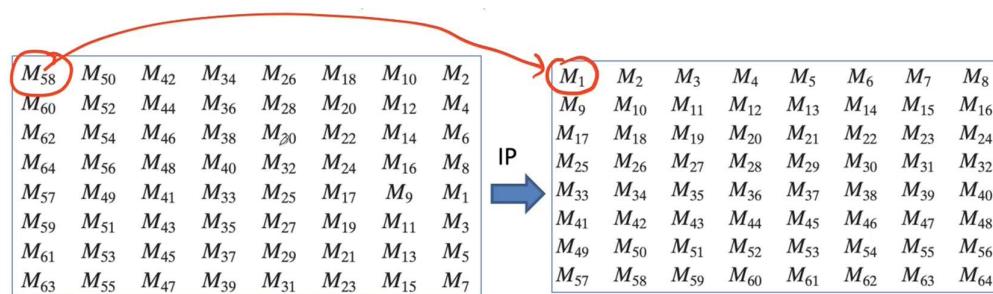
Dữ liệu 64 bit đầu vào được đưa vào bảng hoán vị khởi tạo (*Initial permutation*) sau đó được đưa vào *Round 1*. Trong hàm vòng sẽ sử dụng các cấu trúc đại số, hàm mã hóa giống nhau tuy nhiên khóa đầu vào sẽ khác nhau, khóa này được tạo ra từ hàm sinh khóa, hàm hoán vị, hàm vòng sẽ thực hiện 16 lần đến đầu ra của hàm vòng thứ 16 sẽ thực hiện hoán vị - vị trí (*32-bit swap*) và cuối cùng sẽ được đi vào hàm nghịch đảo hoán vị khởi tạo (*Inverse initial permutation*) để cho ra *64-bit ciphertext*. Trong hàm sinh khóa con sẽ có một hàm dịch vòng trái (*Left circular shift*) sau đó đầu ra sẽ được đi vào bảng hoán vị (*Permuted choice 2*), các bảng hoán vị này đều là các bảng cố định, tham

chiếu và các thuật toán trong hàm vòng, giá trị cố định trong bảng hoán vị đều sẽ được công khai

- Hoán vị khởi tạo (*Initial permutation*)

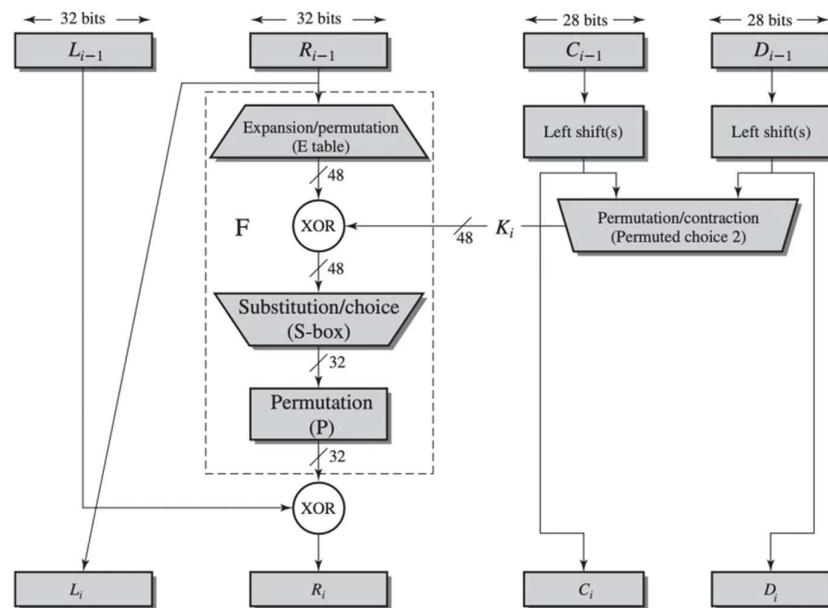
IP ánh xạ 64 bit đầu vào thành 64 bit đầu ra theo một bảng hoán vị cố định, được định nghĩa trong tiêu chuẩn DES và trong bảng hoán vị sẽ chỉ đổi vị trí của các bit chứ không thay đổi giá trị.

Ví dụ:  $X_{58} = IP(M_1)$



Hình 14. Initial permutation

- Hàm vòng (*Round Function*)

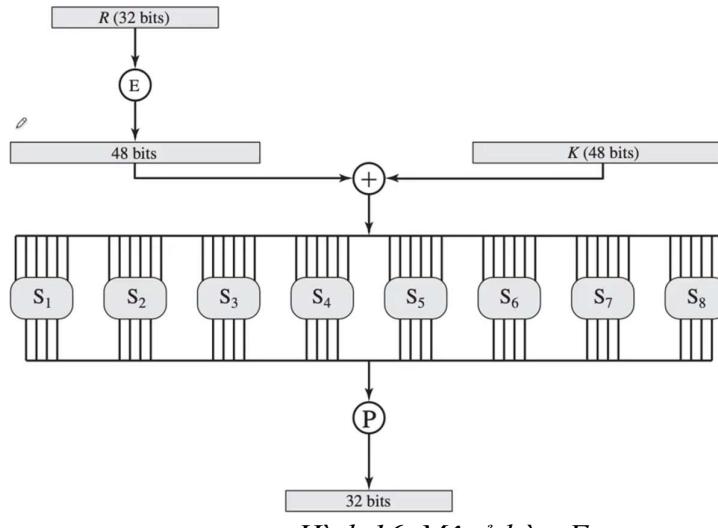


Hình 15. Hàm vòng – Round Function

Đầu vào  $P$  64-bit sẽ bị cắt thành 2 phần  $L_{i-1}$  và  $R_{i-1}$ . Phần  $R_{i-1}$  sẽ trở thành phần  $L_i$  của phần tiếp theo (đây là đặc trưng của mạng Feistel).  $R_{i-1}$  có 32 bit đầu vào bảng mở rộng (*Expansion E*) để mở rộng từ 32-bit thành 48-bit để XOR. Khóa đầu vào 56-bit cũng được chia làm 2 phần, mỗi phần 28-bit và khi đưa vào trong các hàm vòng thì chỉ sử dụng 48-bit để mã hóa (*8-bit còn lại để checksum*), 56 bit được nén thông qua hàm néng (*Permutation/contraction*) tạo ra khóa  $K_i$  có độ dài 48-bit. Trong hàm F thực hiện XOR nhận 2 giá trị đầu vào  $R_{i-1}$  sau khi qua E table và khóa  $K_i$  sẽ tạo ra đầu ra độ dài 48-bit được đưa vào S-box. Trong S-Box có vai trò néng từ 48-bit đầu vào thành 32-bit đầu ra và 32 bit đầu ra này sẽ được hoán vị các vị trí phần tử (*Permutation P*) sẽ cho output đầu ra là 32 bit và lại tiếp tục được XOR với  $L_{i-1}$  cho kết quả là  $R_i$  mô tả như công thức:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \end{aligned}$$

- Hàm F



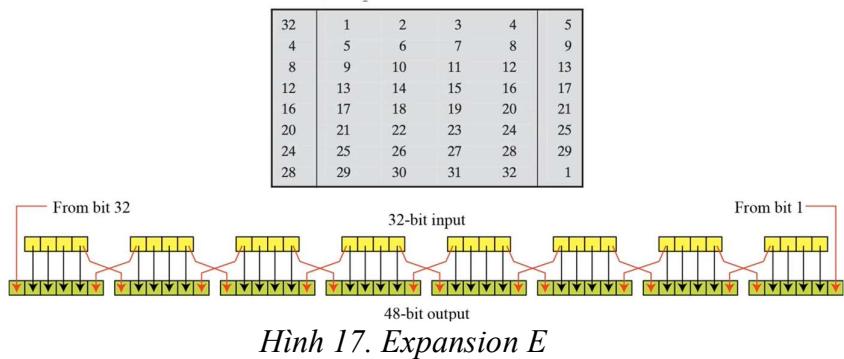
Hình 16. Mô tả hàm F

Đầu vào là  $E(R_{i-1})$  48-bit &  $K_i$  và đầu ra tương ứng 48 bit được chia vào các S-box, kích thước đầu vào mỗi S-box là 6-bit sau đó sẽ được néng lại và có đầu ra 4-bit. Sau đó hợp thành một vector/ma trận 2 chiều được đưa vào một hoán vị  $P$  cho đầu ra 32 bits, cho kết quả là  $R_i$

- Hoán vị mở rộng E (*Expansion E*)

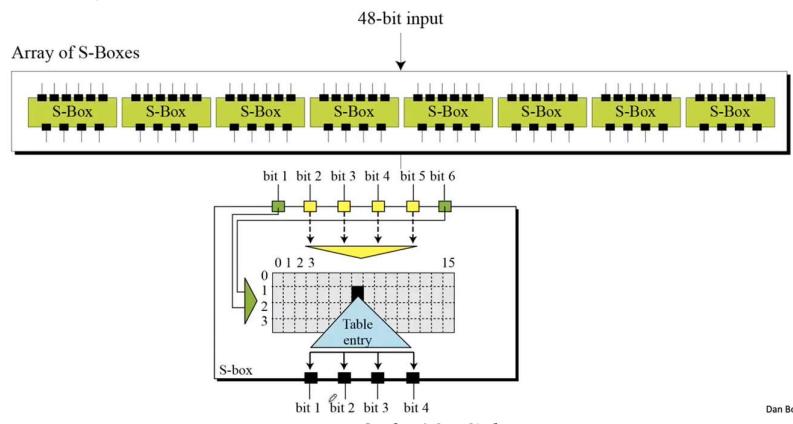
Mở rộng 32-bit đầu vào của  $R_{i-1}$  thành 48 bit đầu ra qua hoán vị mở rộng bằng cách ghép bổ sung các bit theo vị trí của bảng hoán vị mở rộng E.

Ví dụ: Input 1-2-3-4 sẽ được thêm bit 32 (bit cuối của cụm cuối) và bit 5 (bit đầu của cụm kế tiếp) vào vị trí đầu và cuối tương ứng của cụm đầu vào, cho ra output 32-1-2-3-4-5, tương ứng cụm 5-6-7-9 sẽ có output là 4-5-6-7-8-9,...



- Thành phần S-box

Trong S-box có 6 bit đầu vào và sử dụng bảng tham chiếu để nén tạo ra 4 bit đầu ra  $S_i: \{0,1\}^6 \rightarrow \{0,1\}^4$ . Tách bit số 1 và bit số 6, hai bit này sẽ đóng vai trò vị trí theo chiều ngang – vị trí hàng của bảng lookup và 4 bit còn lại là vị trí cột từ đó hình thành bảng  $4 \times 16$  bao gồm 4 hàng và 16 cột



Ví dụ: Trong S-box 5 có  $011011 \rightarrow 1001$

		Middle 4 bits of input															
		S <sub>5</sub>				Outer bits											
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111				Dan Bi											
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001	
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110	
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110	
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0101	0011	0000	

*Hình 19. Ví dụ S-box 5*

S-box 1:																S-box 5:																
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	15	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
S-box 2:																S-box 6:																
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	12	1	10	5	6	1	13	14	0	11	3	8	9	14	15	11	6
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
S-box 3:																S-box 7:																
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
S-box 4:																S-box 8:																
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Hình 20. Bộ đầy đủ 8 S-box theo DES

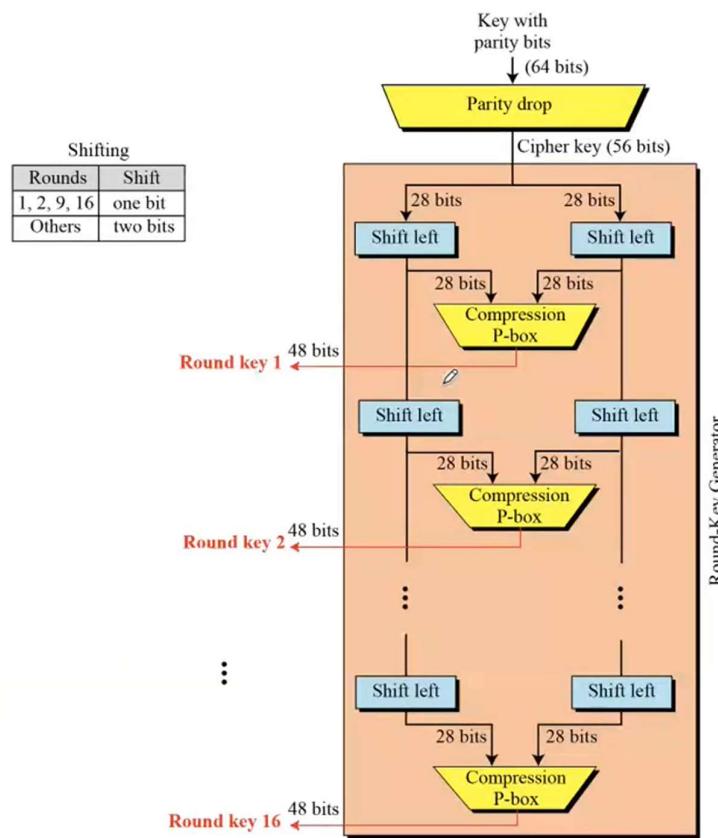
- Hoán vị P (*Straight Permutation*)

Dãy 32 bit đầu ra của khối S-box được sắp xếp lại vị trí (không thay đổi giá trị) theo bảng P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Hình 21. Bảng hoán vị P

- Hàm sinh khóa



Hình 22. Hàm sinh khóa

Bộ tạo khóa vòng tạo ra 16 khóa 48-bit trong bộ mã 56 bit. Trong bộ sinh khóa con được chia làm 2 phần 28 bit. Mỗi phần sẽ được dịch vòng trái 1 bit hoặc 2 bit: vòng 1,2,9,16 sẽ dịch 1 bit, các vòng còn lại dịch 2 bit. Hai phần 28 bit sau khi dịch trái sẽ được đưa vào hộp P (*compression P-box*) để nén tạo khóa 48-bit

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Hình 23. Input key

Bỏ các bit số 0, 8, 16, 24, 32, 40, 48, 56 và 64 (*các vị trí chia hết cho 4*) từ 64 bit khóa ban đầu để loại bỏ 8 bit checksum và thu được 56 bit khóa. Đồng thời sau đó sẽ tự động chia 56 bit thành 2 phần bằng nhau và có độ dài 28 bit

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Hình 24. Permuted Choice One (PC-1)

Sau đó thực hiện phép hoán vị 28 bit đầu vào theo bảng trên để thực hiện quá trình tạo khóa cho 16 vòng (*nửa trên cho phần trái và nửa dưới cho phần phải*)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Hình 25. Permuted Choice Two (PC-2)

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	1	1

Hình 26. Schedule of Left Shifts

Thực hiện ghép 28 bit sau khi hoán vị ở PC-1 thành 56 bit đầu vào và qua hoán vị nén thực hiện theo bảng PC-2 và dịch trái theo quy ước ở trên ta sẽ sinh ra được khóa  $K_i$

Ví dụ:

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Round		$\delta$
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845	1
2	bad228459e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca5e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653c402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		$\delta$
9	c11bfc09887fb6c6 99f911532eed7d94	32
10	887fb6c6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf595606 d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4 7f6e3cf34bcl1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2ed6	31
16	75e8fd8f25896490 1ce2ed6c365e5f59	32
IP <sup>-1</sup>	da02ce3a89ecac3b 057cde97d7683f2a	32

Round		$\delta$
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845	3
2	bad228459e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94ab7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		$\delta$
9	c11bfc09887fb6c6 548f1de471f64df4	34
10	887fb6c6c600f7e8b 71f64df4279786c	36
11	600f7e8bf595606 4279786c399fd0d	32
12	f596506e738538b8 399fd0d6d208dbb	28
13	738538b8c6a62c4 6d208dbbb93bdeea	33
14	c6a62c4e56b0bd75 b9bd3eaa2d2c3a5f	30
15	56b0bd7575e8fd8f d2c3a5f62765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP <sup>-1</sup>	da02ce3a89ecac3b ee92b50606b62b0b	30

Số bit thay đổi giá trị khi thay đổi giá trị bit thứ 4 trong bản rõ

Số bit thay đổi giá trị khi thay đổi giá trị bit thứ 4 trong giá trị khởi tạo khóa

Dan-Boneh

Hình 27. Thay đổi bản rõ và khóa

Round	$K_i$	$L_i$	$R_i$
<b>IP</b>		5a005a00	3cf03c0f
<b>1</b>	1e030f03080d2930	3cf03c0f	bad22845
<b>2</b>	0a31293432242318	bad22845	99e9b723
<b>3</b>	23072318201d0c1d	99e9b723	0bae3b9e
<b>4</b>	05261d3824311a20	0bae3b9e	42415649
<b>5</b>	3325340136002c25	42415649	18b3fa41
<b>6</b>	123a2d0d04262a1c	18b3fa41	9616fe23
<b>7</b>	021f120b1c130611	9616fe23	67117cf2
<b>8</b>	1c10372a2832002b	67117cf2	c11bfc09
<b>9</b>	04292a380c341f03	c11bfc09	887fbcc6c
<b>10</b>	2703212607280403	887fbcc6c	600f7e8b
<b>11</b>	2826390c31261504	600f7e8b	f596506e
<b>12</b>	12071c241a0a0f08	f596506e	738538b8
<b>13</b>	300935393c0d100b	738538b8	c6a62c4e
<b>14</b>	311e09231321182a	c6a62c4e	56b0bd75
<b>15</b>	283d3e0227072528	56b0bd75	75e8fd8f
<b>16</b>	2921080b13143025	75e8fd8f	25896490
<b>IP<sup>-1</sup></b>		da02ce3a	89ecac3b

Hình 28. Kết quả các giá trị trung gian thuật toán

- DES là một hệ mật mã lý tưởng  $[0,1]^{64} = [0,1]^{64}$  ( $2^{56}$  hàm khả nghịch  $\pi_1, \pi_2, \dots, \pi_{2^{56}}$ ) nếu  $\forall m, c \in \{0,1\}^{64} \exists k \in \{0,1\}^{56}$

$$\Pr(c = \text{DES}(k, m)) \geq 1 - 1/2^{56} \sim 99.5\%$$

DES  $\rightarrow$  PRP secure:  $K_X X \rightarrow X [0,1]^{64} = [0,1]^{64}$

- $\Pr(c_1 = \text{DES}(k_1, m)) \leq 1/2^{64}$
- .....
- $\Pr(c_{2^{56}} = \text{DES}(k_{2^{56}}, m)) \leq 1/2^{64}$

$$\Pr(c_i = \text{DES}(k_1, m) = \text{DES}(k_2, m)) \leq \sum_{i=1}^{2^{56}} \Pr(c_i = \text{DES}(k, m)) \leq 2^{56} \cdot 1/2^{64} = 1/2^{56}$$

$$\Rightarrow \Pr(c = \text{DES}(k, m)) \geq 1 - 1/2^{56} \sim 99.5\%$$

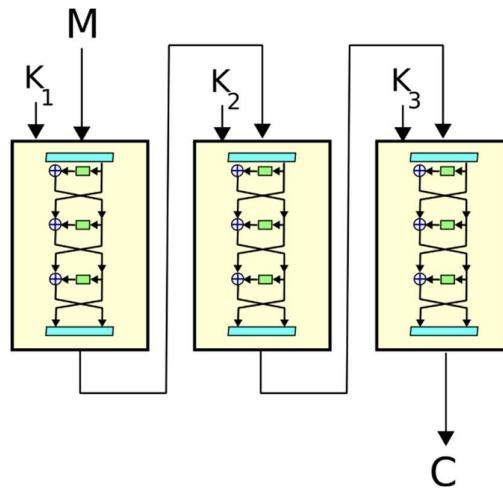
### 4.3. Thuật toán 3DES

3DES (Triple Data Encryption Standard) là một thuật toán mã hóa đối xứng, được phát triển để cải thiện độ an toàn của DES (Data Encryption Standard). Dưới đây là một số điểm nổi bật về 3DES

Nguyên lý hoạt động của 3DES

- Mã hóa ba lần: Như tên gọi, 3DES thực hiện quá trình mã hóa dữ liệu ba lần với ba khóa khác nhau. Điều này làm tăng độ an toàn so với DES chỉ sử dụng một khóa
- Cách thức mã hóa: Dữ liệu được mã hóa bằng DES ba lần, với mỗi lần sử dụng một khóa khác nhau. Cụ thể, quy trình được thực hiện như sau:
  - Mã hóa với khóa 1

- Giải mã với khóa 2
- Mã hóa lại với khóa 3



Hình 29. Sơ đồ hoạt động 3DES

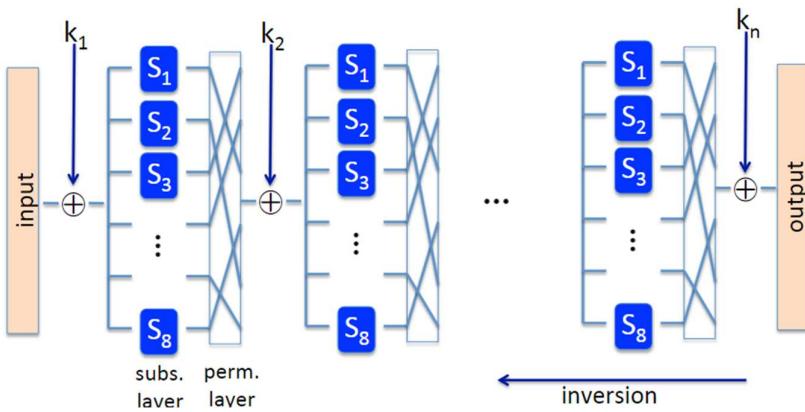
3DES sử dụng ba khóa, mỗi khóa dài 56 bit, dẫn đến tổng độ dài khóa là 168 bit (mặc dù thực tế chỉ có 112 bit độ an toàn do một số vấn đề trong cách tạo khóa). 3DES an toàn hơn DES, cung cấp mức độ bảo mật cao hơn giúp chống lại các cuộc tấn công mã hóa và 3DES cũng được tương thích với các hệ thống cũ đã triển khai trên DES. Tuy nhiên 3DES lại có tốc độ chậm hơn nhiều so với các thuật toán hiện đại như AES do cần phải thực hiện mã hóa 3 lần, gặp khó khăn trong việc quản lý và lưu trữ ba khóa có thể trở nên phức tạp hơn. Mặc dù 3DES vẫn được sử dụng trong một số ứng dụng, nó đang dần bị thay thế bởi các thuật toán mã hóa mạnh mẽ hơn như AES, đặc biệt trong các tiêu chuẩn bảo mật hiện đại.

#### 4.4. Thuật toán AES

##### ➤ Giới thiệu chung về AES

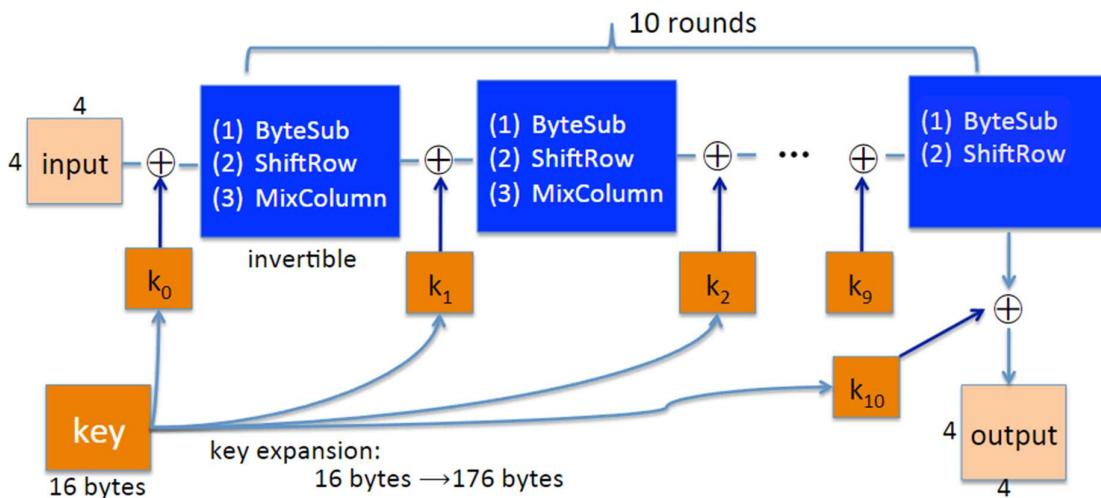
AES (Advanced Encryption Standard) là một thuật toán mã hóa đối xứng được sử dụng rộng rãi để bảo vệ dữ liệu. Nó được phát triển để thay thế cho DES (Data Encryption Standard) và đã trở thành tiêu chuẩn mã hóa chính thức của chính phủ Hoa Kỳ vào năm 2001. AES hỗ trợ ba kích thước khóa: 128, 192 và 256 bit. Kích thước khóa lớn hơn cung cấp mức độ bảo mật cao hơn. AES mã hóa dữ liệu theo khối 128 bit (16 byte) tại một thời điểm và số vòng lặp phụ thuộc vào kích thước của khóa

- 10 vòng cho khóa 128 bit
- 12 vòng cho khóa 192 bit
- 14 vòng cho khóa 256 bit



Hình 30. AES là mạng thay thế hoán vị (Subs-Perm network)

Mạng thay thế hoán vị này gồm lớp thay thế (subs layer) toàn bộ giá trị bit sẽ được thay đổi, vị trí không đổi, hoán vị (perm layer) sẽ thay đổi vị trí các bit và giá trị sẽ không thay đổi. Một khối input có độ dài 128 bit (16 bytes), có khóa độ dài 128 bit, sẽ có 10 vòng, tại mỗi vòng lặp sẽ sử dụng một khóa  $k_i$  bởi hàm sinh khóa.



Hình 31. AES-128

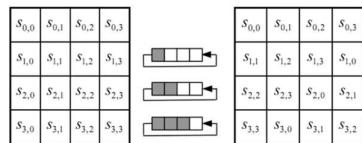
Trong đó:

- (1) ByteSub: Thay đổi về giá trị của byte (thuộc Subs. Layer)
- (2) ShiftRow: Dịch các hàng trong ma trận (thuộc Perm. Layer)
- (3) MixColumns: Trộn cột nằm (thuộc Perm. Layer)
- (4) Addround Key: XOR khóa con

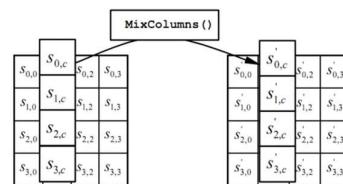
Mỗi một hàm vòng sẽ thực hiện các bước (1)(2)(3) kết hợp (4) và đầu ra của mỗi một hàm vòng sẽ thực hiện 10 vòng đối với AES 128 và tương ứng có 10 khóa con và với hàm vòng cuối cùng (round function 10) sẽ chỉ thực hiện bước (1)(2) và XOR với  $k_{10}$  để cho ra output.

- **ByteSub:** a 1 byte S-box. 256 byte table (easily computable)

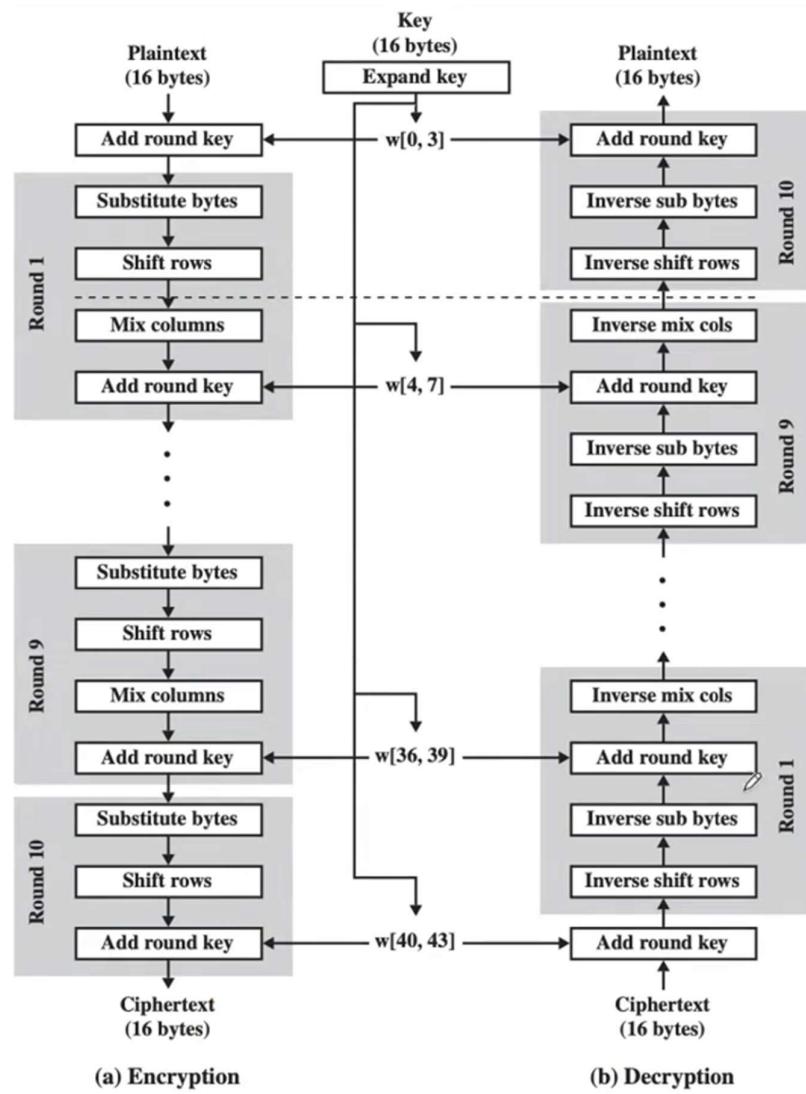
- **ShiftRows:**



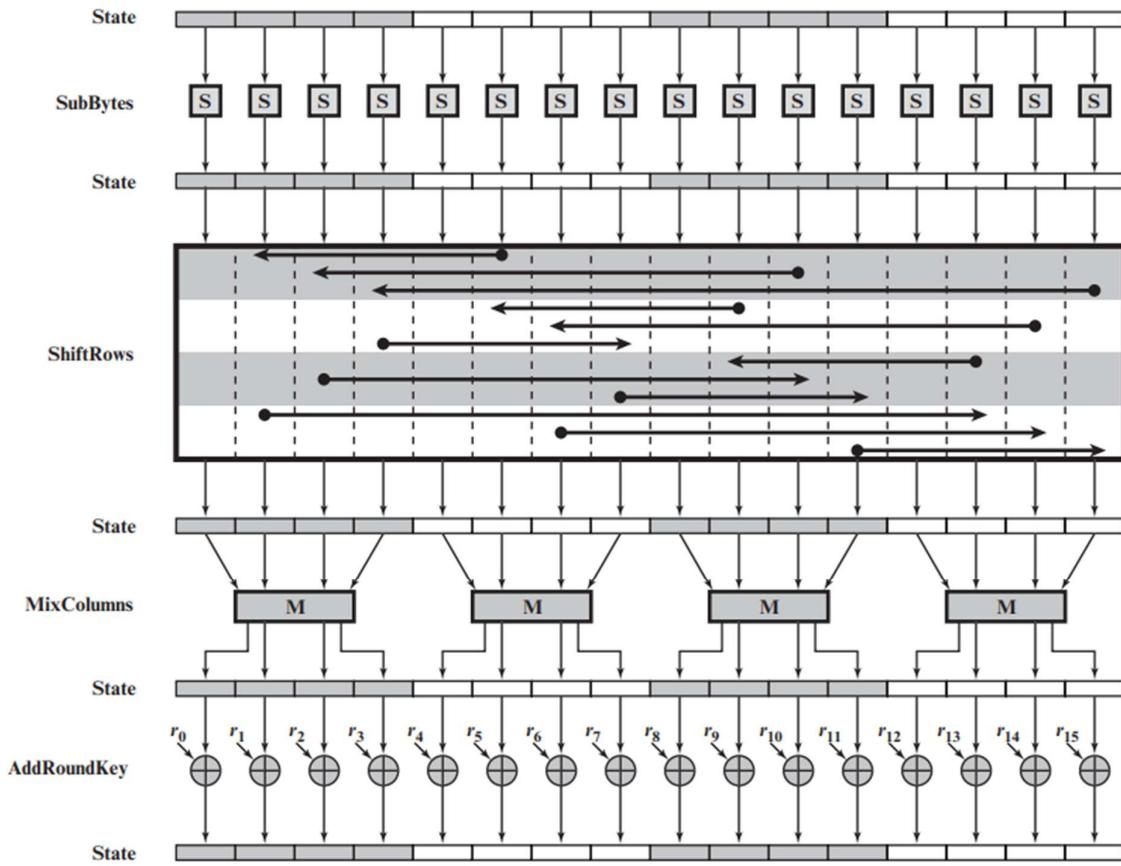
- **MixColumns:**



Hình 32. The round function



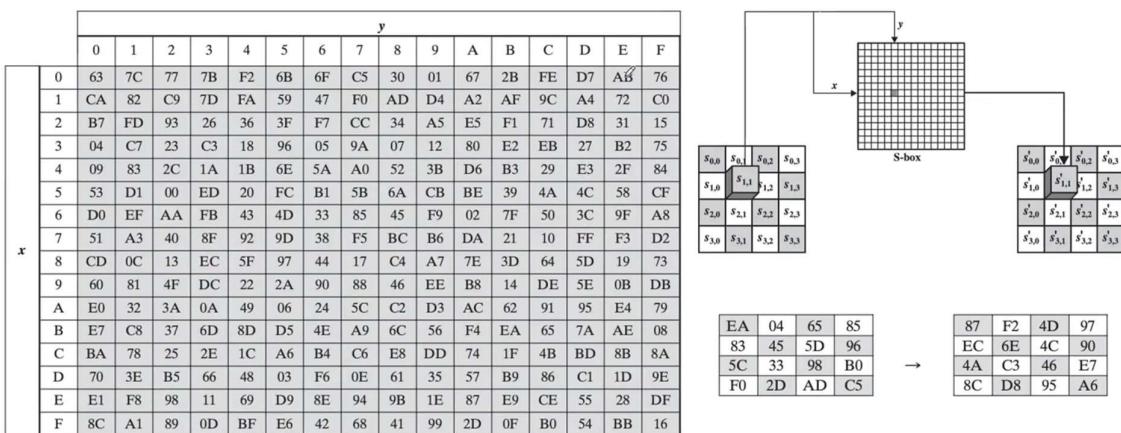
Hình 33. Thuật toán Mã hóa – Giải mã AES



Hình 34. Mô tả chi tiết thuật toán AES

- S: S-box
- M: Ma trận MDS
- r<sub>0</sub>,...,r<sub>15</sub>: Khóa con

DES có 8 S-box, có 8 bảng để biểu diễn giá trị của các S-box, còn đối với AES có 16 S-box nhưng chỉ có 1 bảng S-box 8x8 duy nhất để biểu diễn các giá trị của S-box

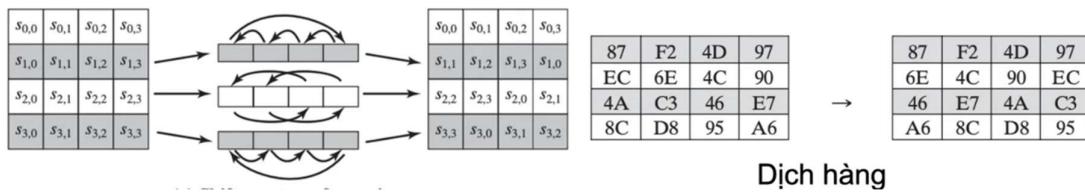


Hình 35. Substitute Bytes Transformation

S-box đầu vào một chuỗi 8 bit, chuỗi 8 bit này được viết lại theo byte tương ứng bit cao, bit thấp (*tùy theo kiểu quy ước địa chỉ bộ nhớ*), ví dụ 1110 1010 thì được viết lại là EA

## IT3150 – Project 1

$(S_{xy})$  tương ứng xét trên bảng tra  $x = E, y = A$  tương ứng sẽ cho ra kết quả đầu ra là 87. Tương ứng với các S-box khác sẽ cho ra kết quả đầu ra (state) để thực hiện ShiftRows. Thực hiện dịch hàng theo hướng dẫn rồi cho output là một state.



Hình 36. Shift row transformation

State ở dịch vòng sẽ được thực hiện phép nhân ma trận với ma trận MDS theo từng  $w_i$  thực hiện Mix column.

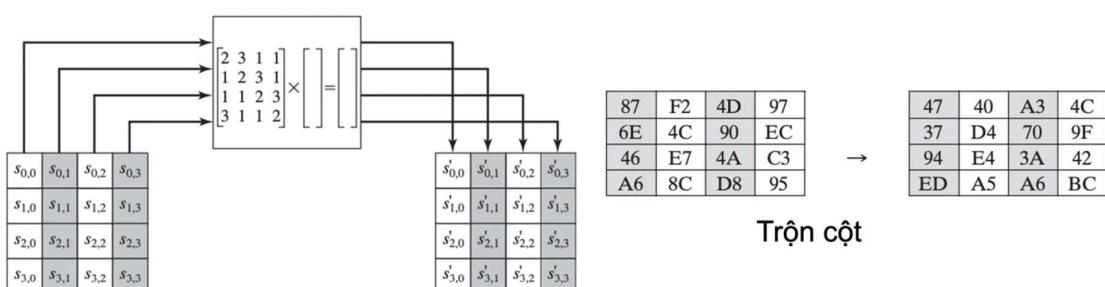
Lưu ý: Nhân ma nhận theo  $GF(2^8)$ ,  $m(x): x^8 - x^4 - x^3 - x - 1$

$w_1$  là 1 word gồm  $s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0}$

$f(x) * g(x)$  trong  $GF(2^8) = x * f(x) \bmod m(x)$ . Trong đó  $f(x) = \{b_7, b_6, b_5, b_4, b_3, b_2, b_1, 0\}$

nếu  $b_7=0$  thì  $x * \{b_6, b_5, b_4, b_3, b_2, b_1, 0\}$

nếu  $b_7=1$  thì  $x * \{b_6, b_5, b_4, b_3, b_2, b_1, 0\} \oplus [x^8 - m(x)]$

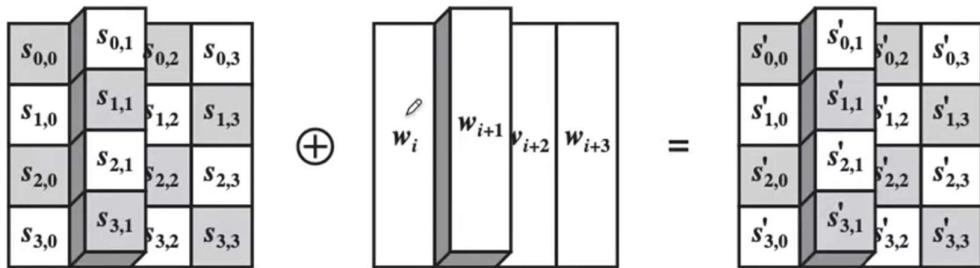


Hình 37. Mix column transformation

Ví dụ:  $\{02\}_{\text{hexa}} * \{87\}_{\text{hexa}} = \{0001\ 1011\} \oplus \{0000\ 1110\} = \{0001\ 0101\}$

$\{01\}_{\text{hexa}} * \{6E\}_{\text{hexa}} = \{0110\ 1110\}$

$\{03\}_{\text{hexa}} * \{6E\}_{\text{hexa}} = \{01\}_{\text{hexa}} * \{6E\}_{\text{hexa}} \oplus \{02\}_{\text{hexa}} * \{6E\}_{\text{hexa}}$



(b) Add round key transformation

47	40	A3	4C	⊕	AC	19	28	57	=	EB	59	8B	1B
37	D4	70	9F		77	FA	D1	5C		40	2E	A1	C3
94	E4	3A	42		66	DC	29	00		F2	38	13	42
ED	A5	A6	BC		F3	21	41	6A		1E	84	E7	D6

Hình 38. Add round key transformation

Mỗi một vòng sẽ được xor với một khóa con được sinh ra từ một hàm sinh khóa (key generator).

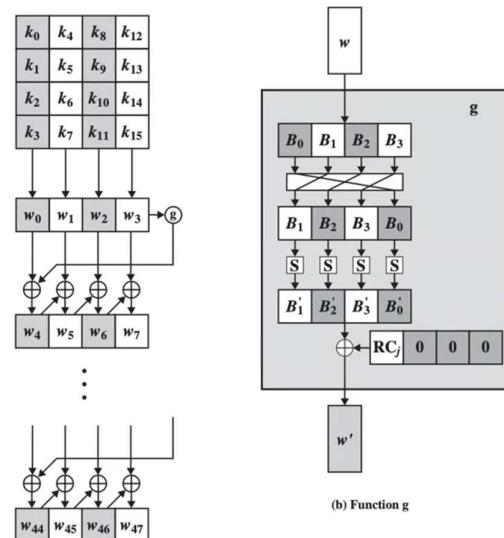
```

KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)
        w[i] = (key[4*i], key[4*i+1],
                 key[4*i+2],
                 key[4*i+3]);

    for (i = 4; i < 44; i++)
    {
        temp = w[i - 1];
        if (i mod 4 = 0)    temp = SubWord (RotWord (temp)
                                         ⊕ Rcon[i/4]);
        w[i] = w[i-4] ⊕ temp
    }
}

```

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36



Hình 39. Key function generator – AES key expansion

Đối với hàm sinh khóa, khóa ban đầu sẽ được biến đổi thành ma trận 2 chiều 4x4 byte và sinh ra các ma trận  $w$  và được đẩy vào trong hàm  $g$ . Mỗi một  $w$  được đưa vào trong hàm  $g$  sẽ được thay đổi vị trí của các byte theo như hình mô tả rồi cho đầu ra vào lại bảng S-box (bảng trong thuật toán tạo mã) sẽ được xor với hàm  $RC[j]$  round counter function(ánh xạ như hình 36)

Ví dụ

Plaintext:	0123456789abcdefedcba9876543210
Key:	0f1571c947d9e8590cb7add6af7f6798
Ciphertext:	ff0b844a0853bf7c6934ab4364148fb9

Key Words	Auxiliary Function
$w_0 = 0f\ 15\ 71\ c9$ $w_1 = 47\ d9\ e8\ 59$ $w_2 = 0c\ b7\ ad\ d6$ $w_3 = af\ 7f\ 67\ 98$	$\text{RotWord}(w_3) = 7f\ 67\ 98\ af = x_1$ $\text{SubWord}(x_1) = d2\ 85\ 46\ 79 = y_1$ $\text{Rcon}(1) = 01\ 00\ 00\ 00$ $y_1 \oplus \text{Rcon}(1) = d3\ 85\ 46\ 79 = z_1$
$w_4 = w_0 \oplus z_1 = dc\ 90\ 37\ b0$ $w_5 = w_4 \oplus w_1 = 9b\ 49\ df\ e9$ $w_6 = w_5 \oplus w_2 = 97\ fe\ 72\ 3f$ $w_7 = w_6 \oplus w_3 = 38\ 81\ 15\ a7$	$\text{RotWord}(w_7) = 81\ 15\ a7\ 38 = x_2$ $\text{SubWord}(x_2) = 0c\ 59\ 5c\ 07 = y_2$ $\text{Rcon}(2) = 02\ 00\ 00\ 00$ $y_2 \oplus \text{Rcon}(2) = 0e\ 59\ 5c\ 07 = z_2$
$w_8 = w_4 \oplus z_2 = d2\ c9\ 6b\ b7$ $w_9 = w_8 \oplus w_5 = 49\ 80\ b4\ 5e$ $w_{10} = w_9 \oplus w_6 = de\ 7e\ c6\ 61$ $w_{11} = w_{10} \oplus w_7 = e6\ ff\ d3\ c6$	$\text{RotWord}(w_{11}) = ff\ d3\ c6\ e6 = x_3$ $\text{SubWord}(x_3) = 16\ 66\ b4\ 83 = y_3$ $\text{Rcon}(3) = 04\ 00\ 00\ 00$ $y_3 \oplus \text{Rcon}(3) = 12\ 66\ b4\ 8e = z_3$
$w_{12} = w_8 \oplus z_3 = c0\ af\ df\ 39$ $w_{13} = w_{12} \oplus w_9 = 89\ 2f\ 6b\ 67$ $w_{14} = w_{13} \oplus w_{10} = 57\ 51\ ad\ 06$ $w_{15} = w_{14} \oplus w_{11} = b1\ ae\ 7e\ c0$	$\text{RotWord}(w_{15}) = ae\ 7e\ c0\ b1 = x_4$ $\text{SubWord}(x_4) = e4\ f3\ ba\ c8 = y_4$ $\text{Rcon}(4) = 08\ 00\ 00\ 00$ $y_4 \oplus \text{Rcon}(4) = ec\ f3\ ba\ c8 = 4$

(Continued)

## IT3150 – Project 1

Key Words	Auxiliary Function
w16 = w12 $\oplus$ z4 = 2c 5c 65 f1 w17 = w16 $\oplus$ w13 = a5 73 0e 96 w18 = w17 $\oplus$ w14 = f2 22 a3 90 w19 = w18 $\oplus$ w15 = 43 8c dd 50	RotWord (w19) = 8c dd 50 43 = x5 SubWord (x5) = 64 c1 53 1a = y5 Rcon(5) = 10 00 00 00 y5 $\oplus$ Rcon(5) = 74 c1 53 1a = z5
w20 = w16 $\oplus$ z5 = 58 9d 36 eb w21 = w20 $\oplus$ w17 = fd ee 38 7d w22 = w21 $\oplus$ w18 = 0f cc 9b ed w23 = w22 $\oplus$ w19 = 4c 40 46 bd	RotWord (w23) = 40 46 bd 4c = x6 SubWord (x6) = 09 5a 7a 29 = y6 Rcon(6) = 20 00 00 00 y6 $\oplus$ Rcon(6) = 29 5a 7a 29 = z6
w24 = w20 $\oplus$ z6 = 71 c7 4c c2 w25 = w24 $\oplus$ w21 = 8c 29 74 bf w26 = w25 $\oplus$ w22 = 83 e5 ef 52 w27 = w26 $\oplus$ w23 = cf a5 a9 ef	RotWord (w27) = a5 a9 ef cf = x7 SubWord (x7) = 06 d3 bf 8a = y7 Rcon(7) = 40 00 00 00 y7 $\oplus$ Rcon(7) = 46 d3 df 8a = z7
w28 = w24 $\oplus$ z7 = 37 14 93 48 w29 = w28 $\oplus$ w25 = bb 3d e7 f7 w30 = w29 $\oplus$ w26 = 38 d8 08 a5 w31 = w30 $\oplus$ w27 = f7 7d a1 4a	RotWord (w31) = 7d a1 4a f7 = x8 SubWord (x8) = ff 32 d6 68 = y8 Rcon(8) = 80 00 00 00 y8 $\oplus$ Rcon(8) = 7f 32 d6 68 = z8
w32 = w28 $\oplus$ z8 = 48 26 45 20 w33 = w32 $\oplus$ w29 = f3 1b a2 d7 w34 = w33 $\oplus$ w30 = cb c3 aa 72 w35 = w34 $\oplus$ w32 = 3c be 0b 3	RotWord (w35) = be 0b 38 3c = x9 SubWord (x9) = ae 2b 07 eb = y9 Rcon(9) = 1b 00 00 00 y9 $\oplus$ Rcon(9) = b5 2b 07 eb = z9
w36 = w32 $\oplus$ z9 = fd 0d 42 cb w37 = w36 $\oplus$ w33 = 0e 16 e0 1c w38 = w37 $\oplus$ w34 = c5 d5 4a 6e w39 = w38 $\oplus$ w35 = f9 6b 41 56	RotWord (w39) = 6b 41 56 f9 = x10 SubWord (x10) = 7f 83 b1 99 = y10 Rcon(10) = 36 00 00 00 y10 $\oplus$ Rcon(10) = 49 83 b1 99 = z10
w40 = w36 $\oplus$ z10 = b4 8e f3 52 w41 = w40 $\oplus$ w37 = ba 98 13 4e w42 = w41 $\oplus$ w38 = 7f 4d 59 20 w43 = w42 $\oplus$ w39 = 86 26 18 76	

(Continued)

Key Words	Auxiliary Function
$w16 = w12 \oplus z4 = 2c\ 5c\ 65\ f1$ $w17 = w16 \oplus w13 = a5\ 73\ 0e\ 96$ $w18 = w17 \oplus w14 = f2\ 22\ a3\ 90$ $w19 = w18 \oplus w15 = 43\ 8c\ dd\ 50$	$\text{RotWord}(w19) = 8c\ dd\ 50\ 43 = x5$ $\text{SubWord}(x5) = 64\ c1\ 53\ 1a = y5$ $Rcon(5) = 10\ 00\ 00\ 00$ $y5 \oplus Rcon(5) = 74\ c1\ 53\ 1a = z5$
$w20 = w16 \oplus z5 = 58\ 9d\ 36\ eb$ $w21 = w20 \oplus w17 = fd\ ee\ 38\ 7d$ $w22 = w21 \oplus w18 = 0f\ cc\ 9b\ ed$ $w23 = w22 \oplus w19 = 4c\ 40\ 46\ bd$	$\text{RotWord}(w23) = 40\ 46\ bd\ 4c = x6$ $\text{SubWord}(x6) = 09\ 5a\ 7a\ 29 = y6$ $Rcon(6) = 20\ 00\ 00\ 00$ $y6 \oplus Rcon(6) = 29\ 5a\ 7a\ 29 = z6$
$w24 = w20 \oplus z6 = 71\ c7\ 4c\ c2$ $w25 = w24 \oplus w21 = 8c\ 29\ 74\ bf$ $w26 = w25 \oplus w22 = 83\ e5\ ef\ 52$ $w27 = w26 \oplus w23 = cf\ a5\ a9\ ef$	$\text{RotWord}(w27) = a5\ a9\ ef\ cf = x7$ $\text{SubWord}(x7) = 06\ d3\ bf\ 8a = y7$ $Rcon(7) = 40\ 00\ 00\ 00$ $y7 \oplus Rcon(7) = 46\ d3\ df\ 8a = z7$
$w28 = w24 \oplus z7 = 37\ 14\ 93\ 48$ $w29 = w28 \oplus w25 = bb\ 3d\ e7\ f7$ $w30 = w29 \oplus w26 = 38\ d8\ 08\ a5$ $w31 = w30 \oplus w27 = f7\ 7d\ a1\ 4a$	$\text{RotWord}(w31) = 7d\ a1\ 4a\ f7 = x8$ $\text{SubWord}(x8) = ff\ 32\ d6\ 68 = y8$ $Rcon(8) = 80\ 00\ 00\ 00$ $y8 \oplus Rcon(8) = 7f\ 32\ d6\ 68 = z8$
$w32 = w28 \oplus z8 = 48\ 26\ 45\ 20$ $w33 = w32 \oplus w29 = f3\ 1b\ a2\ d7$ $w34 = w33 \oplus w30 = cb\ c3\ aa\ 72$ $w35 = w34 \oplus w32 = 3c\ be\ 0b\ 3$	$\text{RotWord}(w35) = be\ 0b\ 38\ 3c = x9$ $\text{SubWord}(x9) = ae\ 2b\ 07\ eb = y9$ $Rcon(9) = 1B\ 00\ 00\ 00$ $y9 \oplus Rcon(9) = b5\ 2b\ 07\ eb = z9$
$w36 = w32 \oplus z9 = fd\ 0d\ 42\ cb$ $w37 = w36 \oplus w33 = 0e\ 16\ e0\ 1c$ $w38 = w37 \oplus w34 = c5\ d5\ 4a\ 6e$ $w39 = w38 \oplus w35 = f9\ 6b\ 41\ 56$	$\text{RotWord}(w39) = 6b\ 41\ 56\ f9 = x10$ $\text{SubWord}(x10) = 7f\ 83\ b1\ 99 = y10$ $Rcon(10) = 36\ 00\ 00\ 00$ $y10 \oplus Rcon(10) = 49\ 83\ b1\ 99 = z10$
$w40 = w36 \oplus z10 = b4\ 8e\ f3\ 52$ $w41 = w40 \oplus w37 = ba\ 98\ 13\ 4e$ $w42 = w41 \oplus w38 = 7f\ 4d\ 59\ 20$ $w43 = w42 \oplus w39 = 86\ 26\ 18\ 76$	

Hình 40. Key expansion for AES example

Cột bên trái là 4 khóa được tạo cho mỗi vòng, cột bên phải là các kết quả các bước được thực hiện trong việc mở rộng khóa. Tiếp theo Hình 38 sẽ tiến trình thay đổi của state trong mã hóa AES. Cột đầu tiên hiển thị giá trị của State tại thời điểm bắt đầu một vòng. Đối với hàng đầu tiên, State chỉ là cách sắp xếp ma trận của văn bản gốc. Các cột thứ hai, thứ ba và thứ tư hiển thị giá trị của State cho vòng đó sau các phép biến đổi SubBytes, ShiftRows và MixColumns, tương ứng. Cột thứ năm hiển thị khóa vòng. Bạn có thể xác minh rằng các khóa vòng này tương đương với những khóa được hiển thị trong Hình 37. Cột đầu tiên hiển thị giá trị của State kết quả từ phép XOR từng bit của State sau phép MixColumns với khóa vòng cho vòng trước.

## IT3150 – Project 1

Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76 23 ab dc 54 45 cd ba 32 67 ef 98 10				0f 47 0c af 15 d9 b7 7f 71 e8 ad 67 c9 59 d6 98
0e ce f2 d9 36 72 6b 2b 34 25 17 55 ae b6 4e 88	ab 8b 89 35 05 40 7f f1 18 3f f0 fc e4 4e 2f c4	ab 8b 89 35 40 7f f1 05 f0 fc 18 3f c4 e4 4e 2f	b9 94 57 75 e4 8e 16 51 47 20 9a 3f c5 d6 f5 3b	dc 9b 97 38 90 49 fe 81 37 df 72 15 b0 e9 3f a7
65 0f c0 4d 74 c7 e8 d0 70 ff e8 2a 75 3f ca 9c	4d 76 ba e3 92 c6 9b 70 51 16 9b e5 9d 75 74 de	4d 76 ba e3 c6 9b 70 92 9b e5 51 16 de 9d 75 74	8e 22 db 12 b2 f2 dc 92 df 80 f7 c1 2d c5 1e 52	d2 49 de e6 c9 80 7e ff 6b b4 c6 d3 b7 5e 61 c6
5c 6b 05 f4 7b 72 a2 6d b4 34 31 12 9a 9b 7f 94	4a 7f 6b bf 21 40 3a 3c 8d 18 c7 c9 b8 14 d2 22	4a 7f 6b bf 40 3a 3c 21 c7 c9 8d 18 22 b8 14 d2	b1 c1 0b cc ba f3 8b 07 f9 1f 6a c3 1d 19 24 5c	c0 89 57 b1 af 2f 51 ae df 6b ad 7e 39 67 06 c0
71 48 5c 7d 15 dc da a9 26 74 c7 bd 24 7e 22 9c	a3 52 4a ff 59 86 57 d3 f7 92 c6 7a 36 f3 93 de	a3 52 4a ff 86 57 d3 59 c6 7a f7 92 de 36 f3 93	d4 11 fe 0f 3b 44 06 73 cb ab 62 37 19 b7 07 ec	2c a5 f2 43 5c 73 22 8c 65 0e a3 dd f1 96 90 50
f8 b4 0c 4c 67 37 24 ff ae a5 c1 ea e8 21 97 bc	41 8d fe 29 85 9a 36 16 e4 06 78 87 9b fd 88 65	41 8d fe 29 9a 36 16 85 78 87 e4 06 65 9b fd 88	2a 47 c4 48 83 e8 18 ba 84 18 27 23 eb 10 0a f3	58 fd 0f 4c 9d ee cc 40 36 38 9b 46 eb 7d ed bd
72 ba cb 04 1e 06 d4 fa b2 20 bc 65 00 6d e7 4e	40 f4 1f f2 72 6f 48 2d 37 b7 65 4d 63 3c 94 2f	40 f4 1f f2 6f 48 2d 72 65 4d 37 b7 2f 63 3c 94	7b 05 42 4a 1e d0 20 40 94 83 18 52 94 c4 43 fb	71 8c 83 cf c7 29 e5 a5 4c 74 ef a9 c2 bf 52 ef
0a 89 c1 85 d9 f9 c5 e5 d8 f7 f7 fb 56 7b 11 14	67 a7 78 97 35 99 a6 d9 61 68 68 0f b1 21 82 fa	67 a7 78 97 99 a6 d9 35 68 0f 61 68 fa b1 21 82	ec 1a c0 80 0c 50 53 c7 3b d7 00 ef b7 22 72 e0	37 bb 38 f7 14 3d d8 7d 93 e7 08 a1 48 f7 a5 4a
db a1 f8 77 18 6d 8b ba a8 30 08 4e ff d5 d7 aa	b9 32 41 f5 ad 3c 3d f4 c2 04 30 2f 16 03 0e ac	b9 32 41 f5 3c 3d f4 ad 30 2f c2 04 ac 16 03 0e	b1 1a 44 17 3d 2f ec b6 0a 6b 2f 42 9f 68 f3 b1	48 f3 cb 3c 26 1b c3 be 45 a2 aa 0b 20 d7 72 38
f9 e9 8f 2b 1b 34 2f 08 4f c9 85 49 bf bf 81 89	99 1e 73 f1 af 18 15 30 84 dd 97 3b 08 08 0c a7	99 1e 73 f1 18 15 30 af 97 3b 84 dd a7 08 08 0c	31 30 3a c2 ac 71 8c c4 46 65 48 eb 6a 1c 31 62	fd 0e c5 f9 0d 16 d5 6b 42 e0 4a 41 cb 1c 6e 56
cc 3e ff 3b a1 67 59 af 04 85 02 aa a1 00 5f 34	4b b2 16 e2 32 85 cb 79 f2 97 77 ac 32 63 cf 18	4b b2 16 e2 85 cb 79 32 77 ac f2 97 18 32 63 cf		b4 ba 7f 86 8e 98 4d 26 f3 13 59 18 52 4e 20 76
ff 08 69 64 0b 53 34 14 84 bf ab 8f 4a 7c 43 b9				

Hình 41. AES example

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cfffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

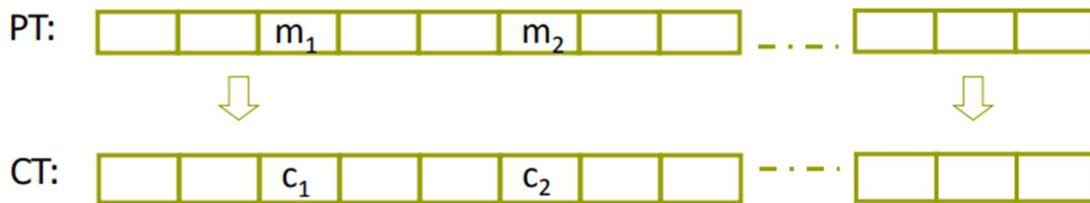
Hình 42. Quá trình thay đổi bản rõ

## 4.5. One Time Key

Giả sử các thuật toán đang sử dụng trong mật mã khối được xây dựng từ một PRP bảo mật, trong hệ thống bảo mật dữ liệu chỉ sử dụng *một khóa cho một lần lần mã hóa* và kẻ tấn công có thể có được một ciphertext từ một lần mã hóa ứng với một khóa mã hóa sử dụng 1 lần và không biết thông tin về khóa. Mục tiêu của kẻ tấn công đó là tìm các thuộc tính liên quan tới *plaintext* từ nhiều *ciphertext* các đặc trưng thống kê từ các ciphertext sinh ra từ hệ thống để suy ngược ra được khóa. Vì vậy các hệ thống sử dụng khóa một lần (*one time key*) cần phải đảm bảo về mặt ngữ nghĩa (*semantic security*)

### 4.5.1. ECB

Electric Code Book (ECB) được mô tả như sau



Hình 43. Hoạt động ECB

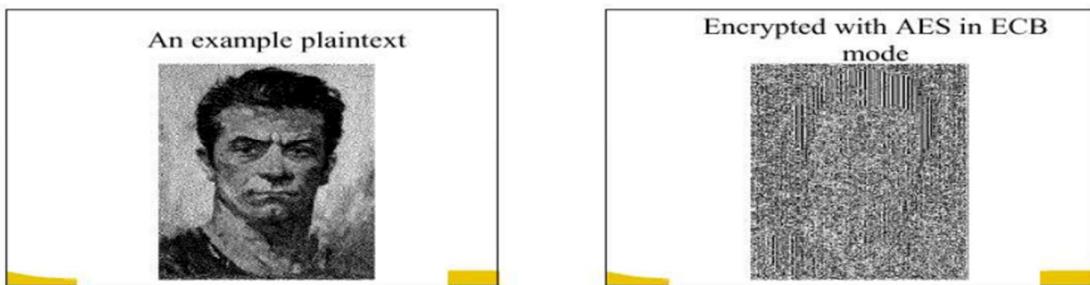
## IT3150 – Project 1

Nếu chiều dài của plaintext có chiều dài hơn 1 khối 128bit thì sẽ được chia ra thành các khối có 128bit và mỗi 1 block sẽ được mã hóa cùng một thuật toán mã khôi cùng với một khóa giống nhau

$$c_1 = E(k, m_1) \text{ & } c_2 = E(k, m_2)$$

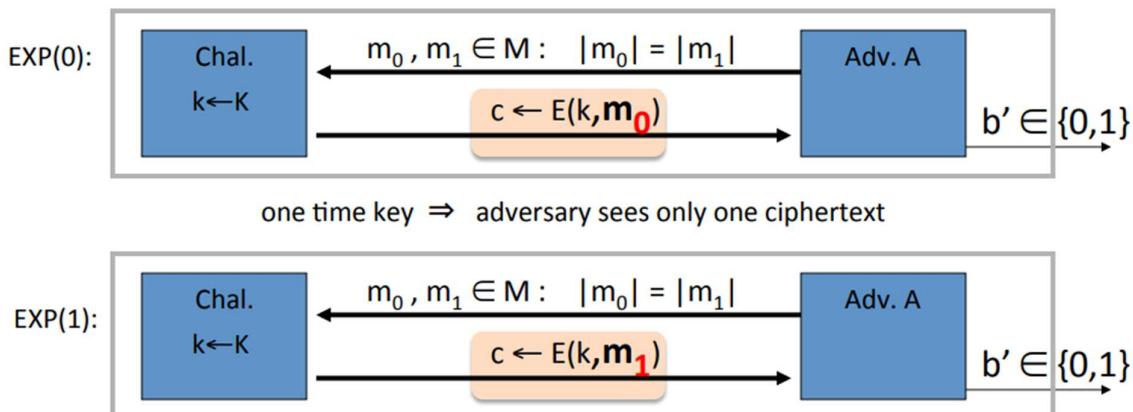
và có một vấn đề nếu như đầu vào  $m_1 = m_2$  thì kết quả đầu ra  $c_1 = c_2$ . Từ đó kẻ tấn công sẽ kiểm tra tần suất các khối lặp lại, các kiểm định thống kê trên các tần suất đấy thì sẽ có thể dự đoán được key. Vậy tại sao ECB lại không bảo mật?

Ví dụ

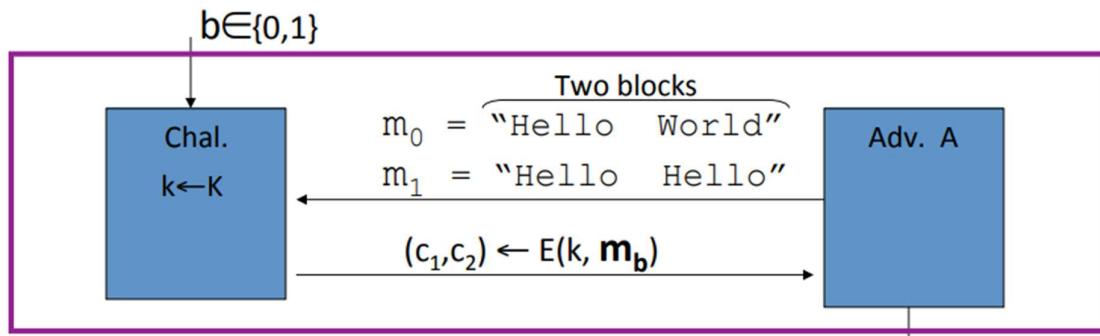


Hình 44. Mã hóa ảnh bằng ECB

Bức ảnh sau khi được mã hóa với AES bằng ECB thì rõ ràng bức ảnh sau khi mã hóa có những điểm trong bản mã là giống nhau, ta có thể dễ dàng nhận ra đường hình mẫu cơ bản của hình gốc ban đầu cho thấy bản mã hóa này không an toàn. Hệ thống an toàn cần điều ra là một xác suất phân phối đều các giá trị đầu ra. Để khóa một lần được đảm bảo tính an toàn thì các hệ thống mã hóa này phải đảm bảo *an toàn ngữ nghĩa* tức là



Lợi thế kẻ tấn công có thể nhận diện được  $m_0$  và  $m_1$  là một số “không đáng kể”. Tuy nhiên một ví dụ khác với ECB



Hình 45. ECB is not Semantically Secure

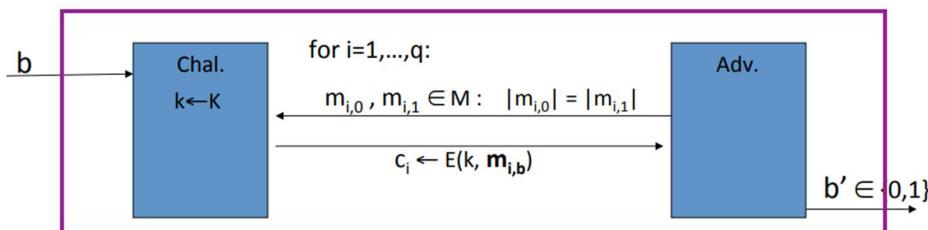
Với đầu vào là 2 plaintext có độ dài bằng nhau  $m_0 = \text{"Hello World"}$  và  $m_1 = \text{"Hello Hello"}$ . Mã hóa  $E(k, m_b)$  với  $b = \{0,1\}$ , đầu vào chia làm 2 khối và trả ra kết quả mã hóa là 2 bản mã hóa, với 2 plaintext trên dễ dàng phân biệt được với trường hợp  $c_1 = c_2$  thì đó là bản mã hóa của plaintext  $m_0$  và  $c_1 \neq c_2$  thì đó là bản mã hóa của plaintext  $m_1$ . Từ đó suy ra lợi thế của kẻ tấn công  $\text{Adv}[ECB] = 1$ . Chứng minh được ECB là một hệ thống không đảm bảo an toàn về mặt ngữ nghĩa

## 4.6. Many Time Key

### 4.6.1. CPA

Chosen-plaintext attack – CPA ứng với khóa nhiều lần là khi kẻ tấn công sẽ có được nhiều ciphertext với cùng một khóa được mã hóa từ các bản tin đầu vào được lựa chọn với mục đích thu thập được nhiều cặp đầu vào và đầu ra  $(m_i, c_i)$  để tìm các mối quan hệ giữa các bản rõ và bản mã này, từ đó có thể khôi phục được các bản rõ và suy ra khóa

$E = (E, D)$  a cipher defined over  $(K, M, C)$ . For  $b = 0, 1$  define EXP(b) as:

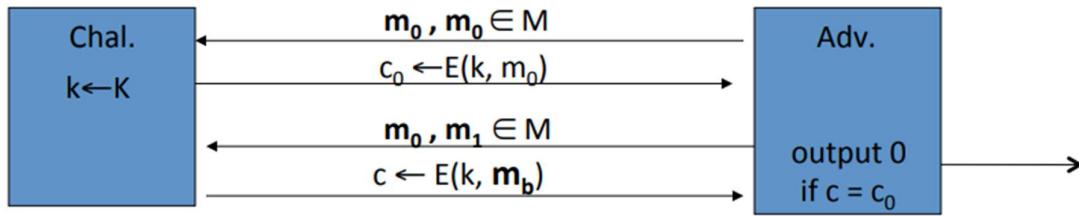


Hình 46. Hoạt động của CPA trong khóa nhiều lần

Cho các đầu vào tương ứng như trong Hình 46, các plaintext được chia thành các khối có kích thước bằng nhau để đưa vào mã hóa và hệ thống trả về các khối ciphertext có kích thước giống nhau. Tuy nhiên nếu có một kiểm định nào đó để nhận diện được

$$c_i \leftarrow E(k, m_{i,b})$$

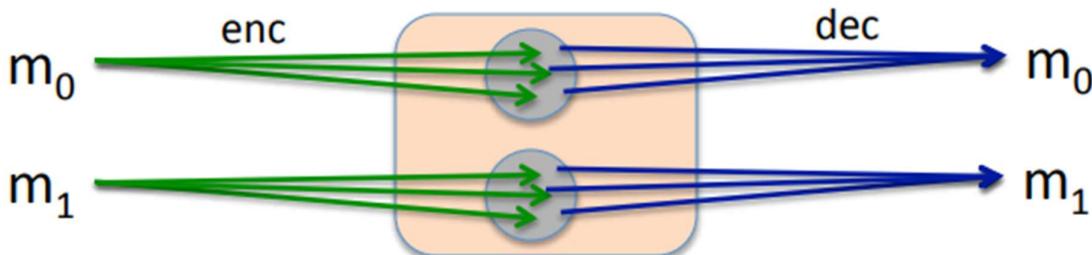
thì kẻ tấn công có thể đếm số lần và tính toán lợi thế của lần tấn công, tương tự với các lần thứ  $i+1$  dẫn tới khả năng hệ thống bảo mật này bị phá vỡ. Tuy nhiên nếu đầu ra của các bản mã hóa  $E(k, m)$  luôn có đầu ra là nhau với cùng bản tin  $m$  thì



Hình 47. Bản mã không bảo mật dưới tấn công CPA

Nếu đầu vào là một bản mã  $m$  có kích thước lớn và các khối giống nhau thì đầu ra sẽ giống nhau và mỗi lần tấn công kẻ tấn công sẽ cho lần thứ nhất là đầu vào giống nhau và lần thứ 2 là khác nhau, tương tự với lần thứ  $n$  thì xác xuất để phân biệt được  $c_0$  và  $c_1$  là bằng 1 và lợi thế tấn công lúc này là đáng kể. Các giải pháp để phòng tránh

Giải pháp 1: Hàm mã hóa ngẫu nhiên



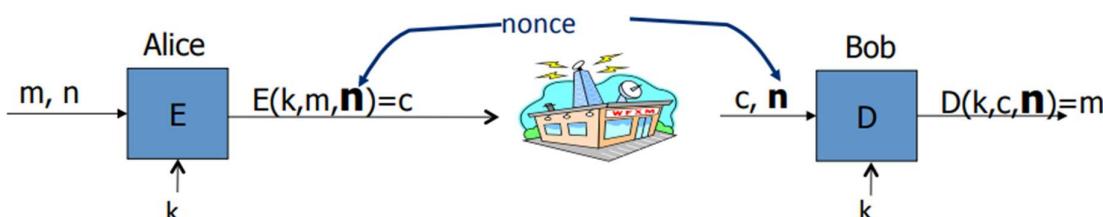
Hình 48. Hàm mã hóa ngẫu nhiên

Sử dụng  $E(k,m)$  là một thuật toán ngẫu nhiên (không phải blockcipher). Lấy đầu ra  $c$  từ các hàm  $E(k,m)$  sẽ cho rất nhiều đầu ra (một hàm toán ánh), đầu vào  $m_0$  thì tạo ra các  $c_i$  khác nhau, vậy để đảm bảo giải mã được trả lại thì đầu vào plaintext cần phải chèn thêm các bit ngẫu nhiên và kích thước của ciphertext là lớn hơn so với plaintext

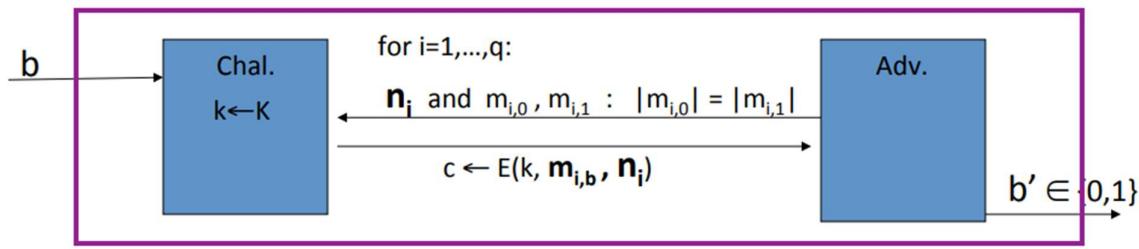
$$CT\text{-size} = PT\text{-size} + \#\text{số bit random}$$

Mô tả qua PRF:  $K \times R \rightarrow M$  với  $m \in M$  và  $E(k,m) = [r \leftarrow R, \text{output } (r, F(k,r) \oplus m)]$ ,  $E$  sẽ là một bảo mật ngữ nghĩa dưới tấn công CPA khi không gian  $R$  đủ lớn để  $r$  không lặp lại

Giải pháp 2: Mã hóa dựa trên số Nonce



*nonce n*: là một giá trị có thể thay đổi từ msg tới msg và cặp  $(k,n)$  không bao giờ được sử dụng hơn một lần. Để đảm bảo an toàn cho hệ thống thì  $n$  được lựa chọn là thật sự ngẫu nhiên hoặc có một không gian  $n$  đủ lớn. Cách lựa chọn đầu tiên nonce là bộ đếm ví dụ bộ đếm  $n_1 = n_0 + 1$ . Cách thứ 2 là một số ngẫu nhiên  $n \leftarrow N$  nhưng với điều kiện  $N$  đủ lớn. Khi suất hiện nonce thì khả năng tấn công

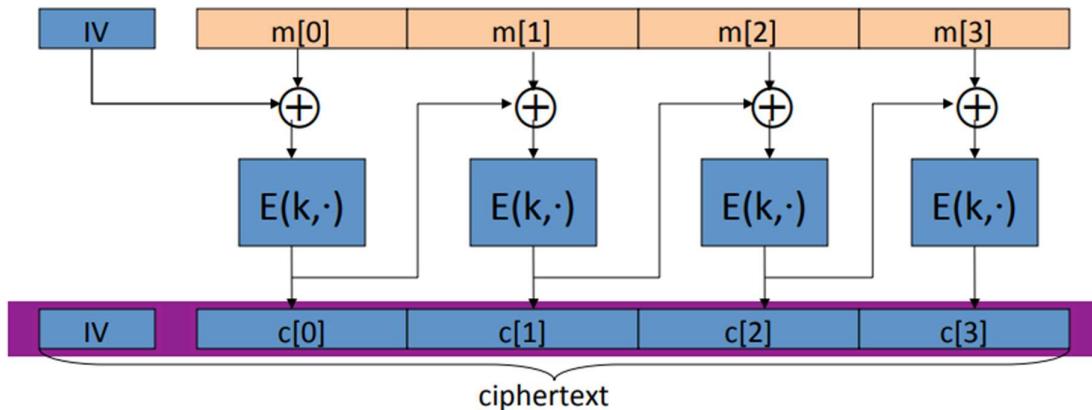


Để phân biệt được *nonce* là của  $m_{i,0}$  nào rất bé với lợi thế tấn công  
 $Adv_{nCPA}[A, E] = |Pr[EXP(0)=1] - Pr[EXP(1)=1]| = "neg"$   
 Phương pháp này được sử dụng các trong các chế độ của mã khối

## 4.7. Các chế độ mã khối

### 4.7.1. CBC

Ta có  $E: K \times [0,1]^n \rightarrow [0,1]^n$  và  $E_{CBC}(k, m)$  chọn một số *nonce* ngẫu nhiên  $IV \in \{0,1\}^n$  và thực hiện



Hình 49. Sơ đồ mã hóa theo chế độ CBC

$$c[0] = E(k, IV \oplus m[0])$$

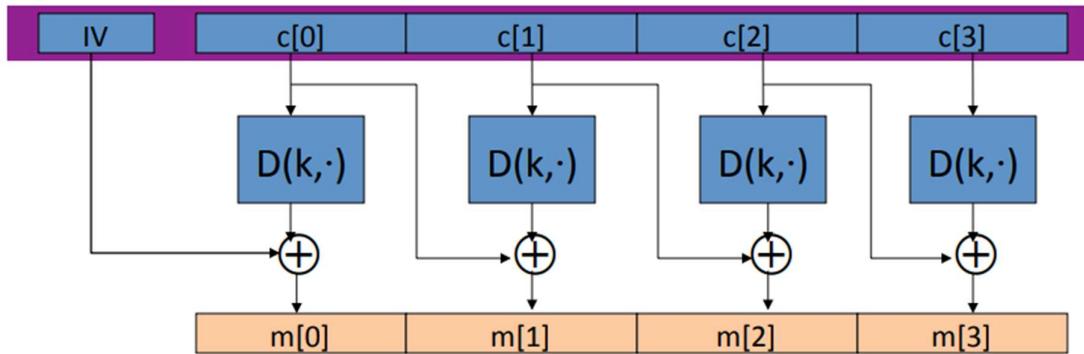
$$c[1] = E(k, c[0] \oplus m[1])$$

$$c[2] = E(k, c[1] \oplus m[2])$$

.....

$(E(k, \cdot)$  là các thuật toán mã khối)

Mỗi lần mã hóa giá trị đầu vào *nonce* là khác nhau và tấn công về mặt ngữ nghĩa không còn khả thi và có thể đảm bảo an toàn. Để giải mã được các ciphertext thì lúc gửi đi cần kèm cả IV bởi vậy ciphertext có chiều dài hơn nhiều so với plaintext do cần phải ghép thêm một số bit random (*bits của IV*). Sau đây là sơ đồ giải mã



Hình 50. Sơ đồ giải mã CBC

$$m[0] = D(k, c[0]) \oplus IV$$

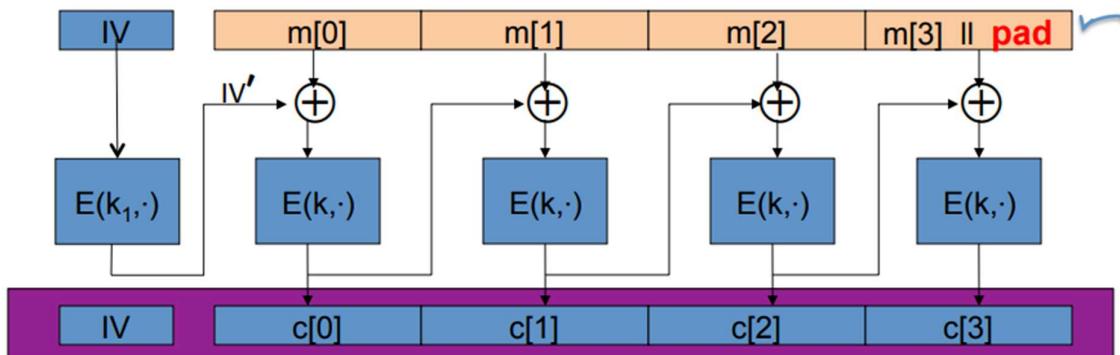
$$m[1] = D(k, c[1]) \oplus c[0]$$

$$m[2] = D(k, c[2]) \oplus c[1]$$

.....

$(D(k, \cdot))$  là các thuật toán giải mã khối)

Kỹ thuật padding cho CBC tăng tính bảo mật. Nếu bản tin đầu vào có chiều dài không chia hết cho số khối thì ta cần *padding* thêm vào cuối của bản tin và lưu lại số bit được thêm vào (khi giải mã thì cần cắt bỏ đi padding). Trong trường hợp nếu block cuối cùng có đủ chiều dài yêu cầu thì ta vẫn cần phải thêm padding vào cuối của bản tin để có thể tăng tính ngẫu nhiên, ví dụ trong Hình 51, nếu  $m[3]$  đủ rồi thì ta thêm khối padding  $m[4]$ . Quy luật padding cần theo byte, ví dụ nếu kích thước block là 8 thì và  $m[3]$  đang có 4 byte thì ta cần padding thêm 4 byte. Để tăng thêm bảo mật và khắc phục *điểm yếu* CBC thì đầu vào thật sự ngẫu nhiên IV sẽ được mã hóa thành  $IV'$  trước khi tham gia vào mã hóa CBC-padding

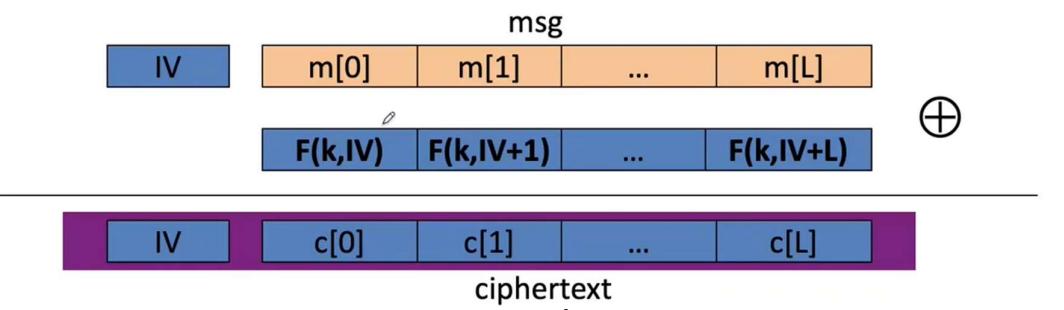


Hình 51. CBC - Padding

#### 4.7.2. CTR

Cấu trúc của Counter mode – CTR bao gồm:

Giả có  $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$  là một PRF. Mã hóa  $E(k,m)$  chọn một  $IV \in \{0,1\}^n$  thật sự ngẫu nhiên, có sơ đồ mã hóa



Hình 52. Sơ đồ mã hóa CTR

Giả định chia  $msg$  thành  $L$  block và mỗi khối có chiều dài  $|x|$ , đầu tiên sẽ mã hóa  $IV$  bằng  $F(k, IV)$  và xor với  $m[0]$ , giá trị  $IV_i = IV_{i-1} + 1$  tương tự như vậy sẽ mã hóa các khối lại. Các phép xor này sẽ được thực hiện đồng thời bởi các  $IV_i$  này được tính từ trước, hiệu năng của CTR mạnh hơn của CBC.

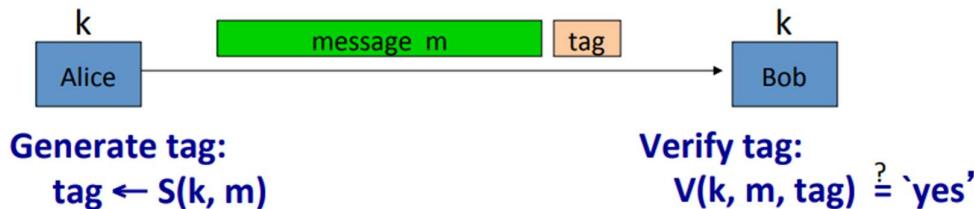
	CBC	ctr mode
uses	PRP	PRF
parallel processing	No	Yes
Security of rand. enc.	$q^2 L^2 \ll  X $	$q^2 L \ll  X $
dummy padding block	Yes	No
1 byte msgs (nonce-based)	16x expansion	no expansion

Hình 53. So sánh CBC & CTR mode

## CHƯƠNG 5. TOÀN VẸN THÔNG ĐIỆP

### 5.1.1. MAC

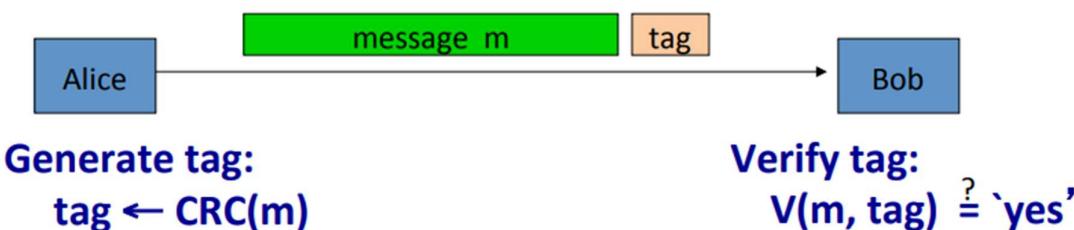
MAC – mã xác thực thông điệp có mục tiêu kiểm tra tính toán vẹn(integrity) và không tính bí mật của bản tin tức là gửi bản tin nhưng không cần dấu bản tin bí mật. Cơ chế cần để kiểm tra bản tin nhận được có phải là bản tin gửi hay không. Ứng dụng bảo vệ các mã nhị phân hay bảo vệ các bảng quảng cáo. Mục tiêu của MAC đó là chống kẻ tấn công không thể tạo tag hợp lệ cho một thông điệp mới



*Hình 54. Message integrity: MACs*

Định nghĩa: MAC  $I = (S, V)$  định nghĩa trên  $(K, M, T)$  là một cặp thuật toán  $S(k, m)$  thuộc  $T$  và  $V(k, m, t)$  output ‘yes’ hoặc ‘no’

Alice gửi cho Bob bản tin kèm theo tag. Cơ chế để tạo ra tag sử dụng hàm  $S(k, m)$  (blackbox). Tag được gắn kèm bản tin gửi đi thì Bob sẽ nhận được cơ chế  $V(k, m, tag)$  là cơ chế kiểm tra bản tin. Đảm bảo tính toàn vẹn thông điệp cần có một khóa bí mật

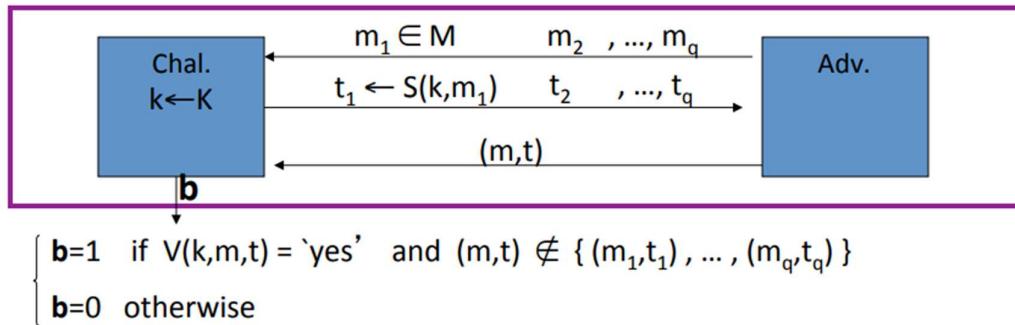


*Hình 55. MACs – Secret Key*

Tag được tạo ra từ CRC (Cyclic redundancy check), CRC được thiết kế để phát hiện lỗi xảy ra ngẫu nhiên chứ không chống được lỗi chủ đích.

Tính bảo mật của MAC: Kẻ tấn công dựa trên CPA đưa ra các truy vấn  $m_1, m_2, \dots$  và nhận được  $t_i \leftarrow S(k, m_i)$ . Dựa trên các cặp msg/tag  $(m, t)$  hợp lệ để gửi tới bên nhận rằng cặp giả mạo  $(m, t)$  được bên nhận trả về giá trị *true*. Vậy mục tiêu của MAC bảo mật cần đảm bảo rằng ngăn chặn được các tag không hợp lệ và từ các  $(m, t)$  hợp lệ không thể tạo ra được  $(m, t')$  cũng hợp lệ với  $t \neq t'$

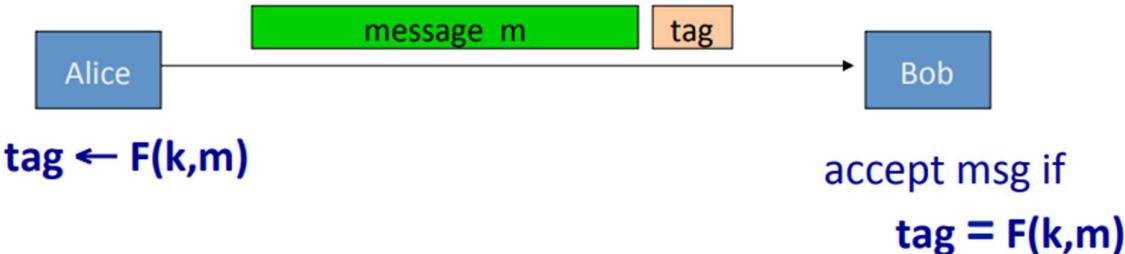
- For a MAC  $I = (S, V)$  and adv. A define a MAC game as:



MAC  $I = (S, V)$  là MAC bảo mật nếu với mọi thuật toán hiệu quả “A” thì  $\text{Adv}_{\text{MAC}}[A, I] = \Pr[\text{output} = 1]$  là “không đáng kể”

### 5.1.2. MAC dựa trên PRF

Xét PRF  $F: K \times X \rightarrow Y$ , định nghĩa MAC:  $I_F = (S, V)$  bởi  $S(k, m) = F(k, m)$  và  $V(k, m, t) = [\text{'yes' nếu } t = F(k, m) - \text{'no' nếu } t \neq F(k, m)]$



Hình 56. MACs – PRF

Nếu  $F(k, \cdot)$  là một PRF bảo mật và  $1/|Y|$  là không đáng kể thì  $I_F$  sẽ là một MAC bảo mật tức là với mọi thuật toán hiệu quả A tấn công MAC  $I_F$  có tồn tại một thuật toán hiệu quả B tấn công PRF  $F$  thỏa mãn

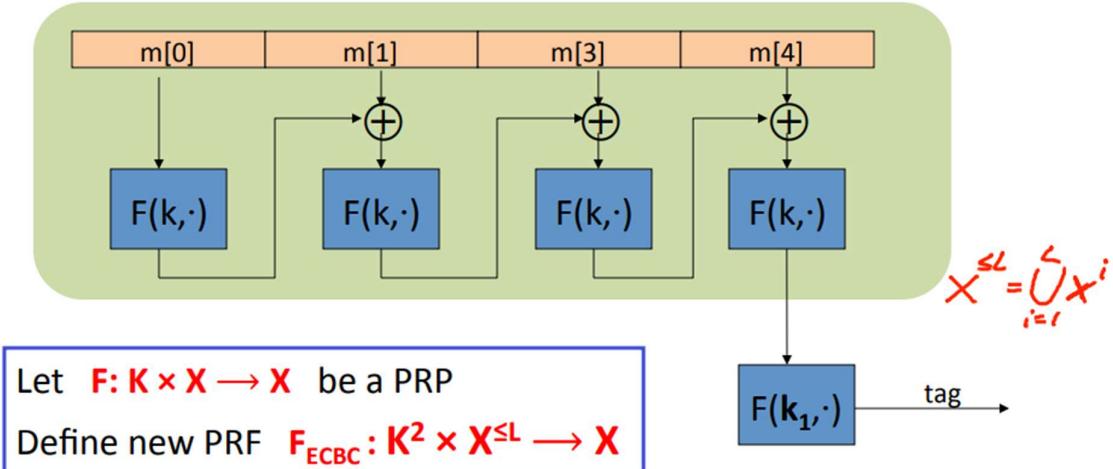
$$\text{Adv}_{\text{MAC}}[A, I_F] \leq \text{Adv}_{\text{PRF}}[B, F] + 1/|Y|$$

An toàn khi  $|Y|$  lớn (vd:  $|Y| = 2^{80}$ )

### 5.1.3. CBC-MAC & NMAC

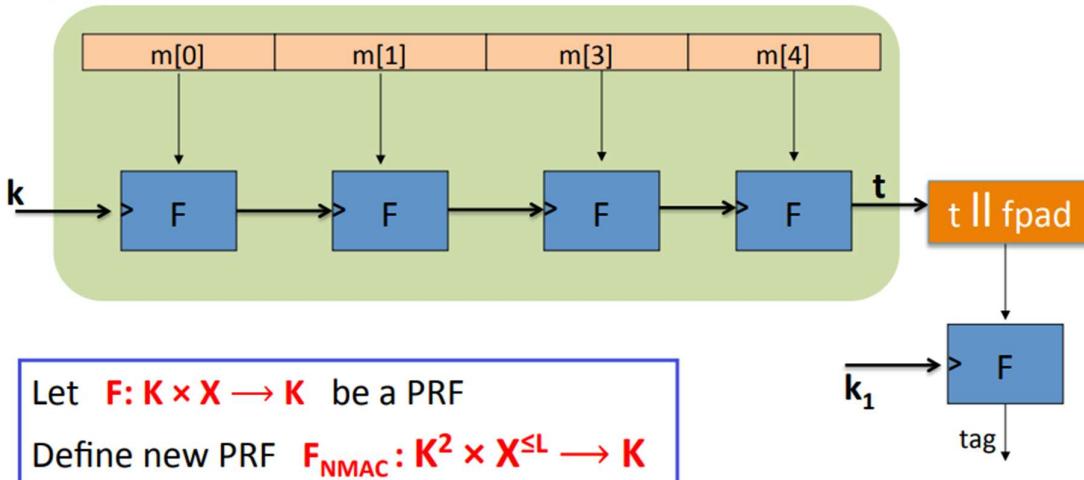
Để là một MAC  $I_F$  bảo mật thì cần xây dựng PRF  $F$  an toàn khi  $|Y|$  lớn và các cách để xây dựng PRF cho một thông điệp đầu vào dài tùy ý

raw CBC



Hình 57. CBC-MAC

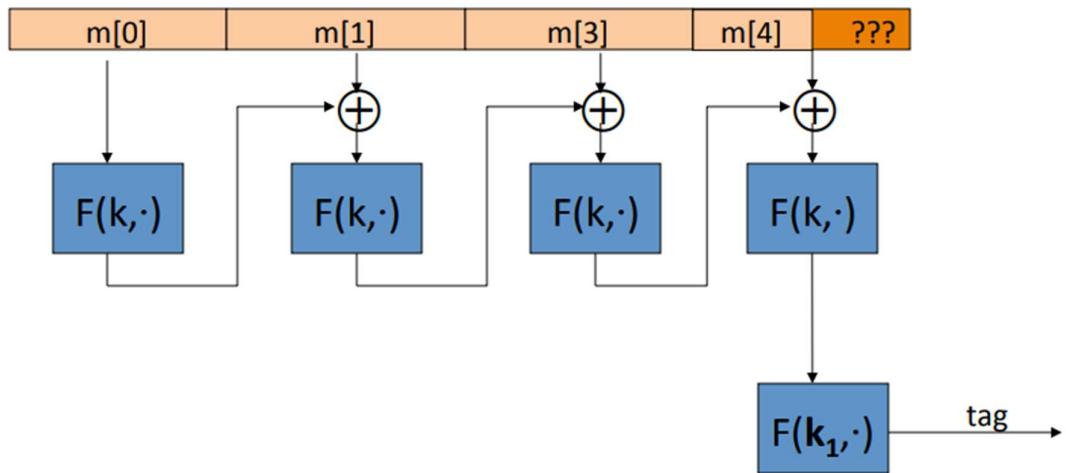
cascade



Hình 58. NMAC

Nếu trong trường hợp thông điệp không là bội của các Block-size thì ta cần phải padding thêm vào các block theo nguyên tắc bit đầu tiên của phần đệm là 1 và các bit còn lại là bit 0 để phân biệt được phần đệm bắt đầu từ đâu “10000...” Và đảm bảo tính chất phần đệm của một chuỗi không được xung đột

$$m_0 \neq m_1 \Rightarrow pad(m_0) \neq pad(m_1)$$



Hình 59. MAC Padding

## CHƯƠNG 6. HÀM BĂM MẬT MÃ

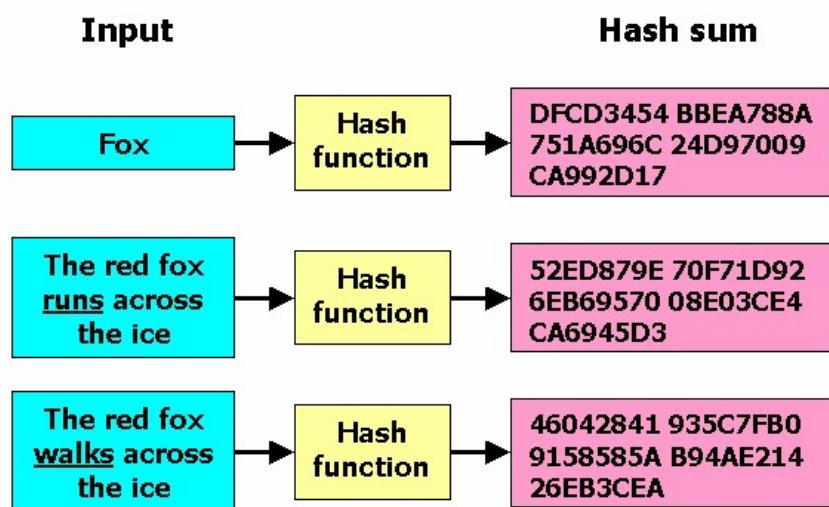
Hàm băm mật mã học (*Cryptographic hash function*) là một hàm băm với một số tính chất bảo mật nhất định để phù hợp việc sử dụng trong nhiều ứng dụng bảo mật thông tin đa dạng, chẳng hạn như chứng thực (authentication) và kiểm tra tính nguyên vẹn của thông điệp (message integrity). Một hàm băm nhận đầu vào là một xâu ký tự dài (hay thông điệp) có độ dài tùy ý và tạo ra kết quả là một xâu ký tự có độ dài cố định, đôi khi được gọi là tóm tắt thông điệp (message digest) hoặc chữ ký số (digital fingerprint). Các hàm băm cần đảm bảo các tính chất sau:

**Đầu vào và đầu ra:** Hàm băm mật mã nhận một chuỗi dữ liệu đầu vào có độ dài bất kỳ và tạo ra một giá trị băm có độ dài cố định.

**Tính duy nhất:** Mỗi đầu vào khác nhau sẽ tạo ra một giá trị băm khác nhau.

**Tính không thể đảo ngược:** Không thể tái tạo lại dữ liệu gốc từ giá trị băm.

**Tính kháng va chạm:** Khó có thể tìm được hai đầu vào khác nhau mà tạo ra cùng một giá trị băm.

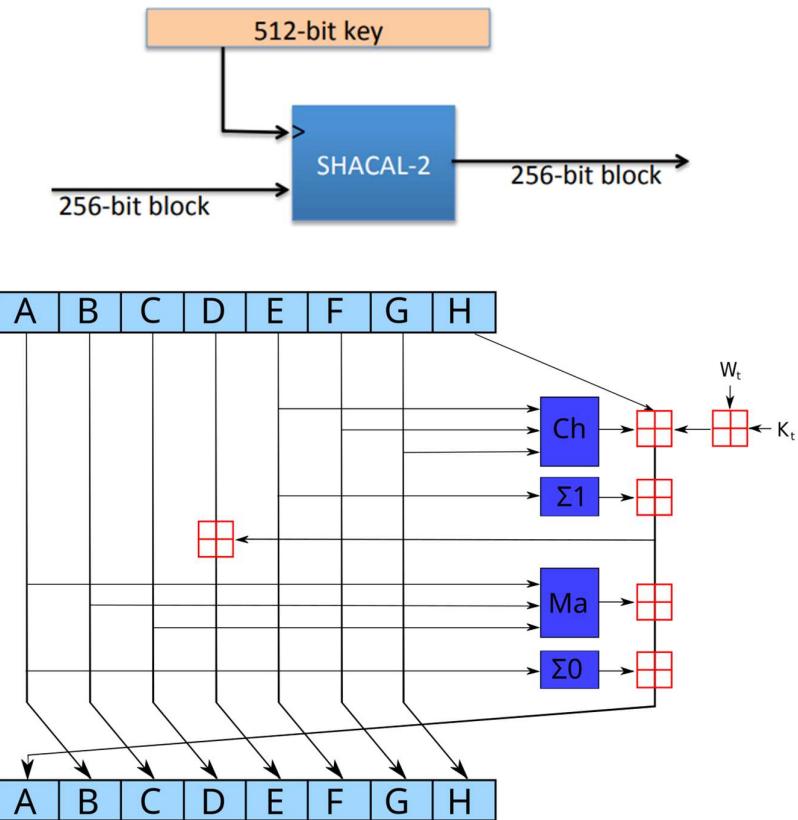


Hình 60. Hoạt động của hàm băm

Trong nhiều chuẩn và ứng dụng, hai hàm băm thông dụng nhất là MD5 và SHA-256. Trong đó hiện nay MD5 không còn được an toàn nhưng là một trong những hàm băm phổ biến và SHA-256 là họ hàm băm SHA-2, được coi là an toàn và được sử dụng rộng rãi

### 6.1.1. Thuật toán hàm băm SHA-256A

SHA-256 (*Secure Hash Algorithm 256-bit*) là một trong những hàm băm mật mã phổ biến nhất, thuộc gia đình SHA-2, được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) và được công bố bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) vào năm 2001.



Hình 61. One iteration in a SHA-2 family compression function

SHA-256 xử lý một thông điệp có độ dài biến đổi thành một đầu ra có độ dài cố định 256 bit. Thông điệp đầu vào được chia thành các khối 512-bit ( $W_t$ ), thông điệp được thêm đệm để chiều dài của nó chia hết cho 512. Thuật toán SHA-256 hoạt động trên trạng thái 256-bit, được chia thành từ 4 đoạn, mỗi đoạn có độ dài 32-bit, với ký hiệu A, B, C, D, E, F, G, H trong đó có giá trị khởi tạo cố định cho các biến băm ở mỗi giai đoạn khác nhau ( $K_i$ ) và được thực hiện lần lượt theo hệ thống *hình 42* với các hàm cho cố định như sau

$$\begin{aligned}
 \text{Ch}(E, F, G) &= (E \wedge F) \oplus (\neg E \wedge G) \\
 \text{Ma}(A, B, C) &= (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C) \\
 \Sigma_0(A) &= (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22) \\
 \Sigma_1(E) &= (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)
 \end{aligned}$$

Trong đó:  $\boxplus$  là phép cộng modulo  $2^{24}$  cho SHA-256

Hiện nay, SHA-256 được coi là an toàn và không có lỗ hổng bảo mật nghiêm trọng nào được phát hiện tính đến thời điểm hiện tại và đang được ứng dụng trong nhiều ứng dụng bảo mật, bao gồm xác thực dữ liệu và lưu mật khẩu, là một phần cốt lõi trong hệ thống *Blockchain* bao gồm Bitcoin, giúp đảm bảo tính toán vẹn, bảo mật dữ liệu giao dịch và được sử dụng trong quy trình tạo chữ ký số.

## Mã giả SHA-256 (*tham khảo\**)

*Note 1: All variables are 32 bit unsigned integers and addition is calculated modulo  $2^{32}$*

*Note 2: For each round, there is one round constant  $k[i]$  and one entry in the message schedule array  $w[i]$ ,  $0 \leq i \leq 63$*

*Note 3: The compression function uses 8 working variables,  $a$  through  $h$*

*Note 4: Big-endian convention is used when expressing the constants in this pseudocode,*

*and when parsing message block data from bytes to words, for example,  
the first word of the input message "abc" after padding is 0x61626380*

### *Initialize hash values:*

(first 32 bits of the *fractional parts* of the square roots of the first 8 primes 2..19):

$h_0 := 0x6a09e667$

$h_1 := 0xbb67ae85$

$h_2 := 0x3c6ef372$

$h_3 := 0xa54ff53a$

$h_4 := 0x510e527f$

$h_5 := 0x9b05688c$

$h_6 := 0x1f83d9ab$

$h_7 := 0x5be0cd19$

### *Initialize array of round constants:*

(first 32 bits of the *fractional parts* of the cube roots of the first 64 primes 2..311):

$k[0..63] :=$

0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1,  
0x923f82a4, 0xab1c5ed5,

0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe,  
0x9bdc06a7, 0xc19bf174,

0xe49b69c1, 0xefbe4786, 0xfc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa,  
0x5cb0a9dc, 0x76f988da,

0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147,  
0x06ca6351, 0x14292967,

0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb,  
0x81c2c92e, 0x92722c85,

0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624,  
0xf40e3585, 0x106aa070,

0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a,  
0x5b9cca4f, 0x682e6ff3,

0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90beffa, 0xa4506ceb,  
0xbef9a3f7, 0xc67178f2

### *Pre-processing (Padding):*

begin with the original message of length L bits

append a single '1' bit

## IT3150 – Project 1

append K '0' bits, where K is the minimum number  $\geq 0$  such that  $(L + 1 + K + 64)$  is a multiple of 512  
append L as a 64-bit big-endian integer, making the total post-processed length a multiple of 512 bits  
such that the bits in the message are: <original message of length L> 1 <K zeros> <L as 64 bit integer>, (the number of bits will be a multiple of 512)

*Process the message in successive 512-bit chunks:*

break message into 512-bit chunks

**for** each chunk

    create a 64-entry message schedule array w[0..63] of 32-bit words

    (*The initial values in w[0..63] don't matter, so many implementations zero them here*)

    copy chunk into first 16 words w[0..15] of the message schedule array

*Extend the first 16 words into the remaining 48 words w[16..63] of the message schedule array:*

**for i from** 16 to 63

        s0 := (w[i-15] **righthrotate** 7) **xor** (w[i-15] **righthrotate** 18) **xor** (w[i-15] **rightshift** 3)  
        s1 := (w[i-2] **righthrotate** 17) **xor** (w[i-2] **righthrotate** 19) **xor** (w[i-2] **rightshift** 10)  
        w[i] := w[i-16] + s0 + w[i-7] + s1

*Initialize working variables to current hash value:*

a := h0

b := h1

c := h2

d := h3

e := h4

f := h5

g := h6

h := h7

*Compression function main loop:*

**for i from** 0 to 63

    S1 := (e **righthrotate** 6) **xor** (e **righthrotate** 11) **xor** (e **righthrotate** 25)

    ch := (e **and** f) **xor** ((**not** e) **and** g)

    temp1 := h + S1 + ch + k[i] + w[i]

    S0 := (a **righthrotate** 2) **xor** (a **righthrotate** 13) **xor** (a **righthrotate** 22)

    maj := (a **and** b) **xor** (a **and** c) **xor** (b **and** c)

    temp2 := S0 + maj

    h := g

    g := f

    f := e

    e := d + temp1

    d := c

## IT3150 – Project 1

```
c := b  
b := a  
a := temp1 + temp2
```

*Add the compressed chunk to the current hash value:*

```
h0 := h0 + a  
h1 := h1 + b  
h2 := h2 + c  
h3 := h3 + d  
h4 := h4 + e  
h5 := h5 + f  
h6 := h6 + g  
h7 := h7 + h
```

*Produce the final hash value (big-endian):*

digest := hash := h0 **append** h1 **append** h2 **append** h3 **append** h4 **append** h5  
**append** h6 **append** h7

Ví dụ: Đầu vào là một chuỗi rỗng “ ”

Input	
Output	0xc672b8d1ef56ed28ab87c3622c5114069bdd3ad7b8f9737498d0c01ecef0967a

## CHƯƠNG 7. MẬT MÃ XÁC THỰC

*Tính bí mật: Bảo mật ngũ ngũa chống được tấn công CPA*

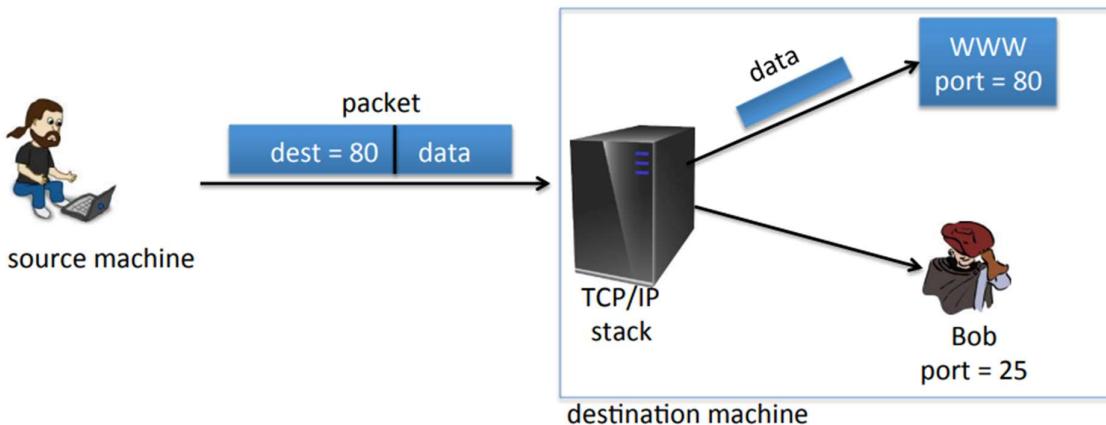
*Tính toàn vẹn: Khả năng không thể giả mạo chống lại các tấn công lựa chọn bản rõ*

*Hệ thống cần đảm bảo đồng thời tính bí mật và toàn vẹn (Authencitaced Encryption)*

### 7.1. Tấn công CPA bảo mật

Ví dụ một cuộc tấn công giả mạo

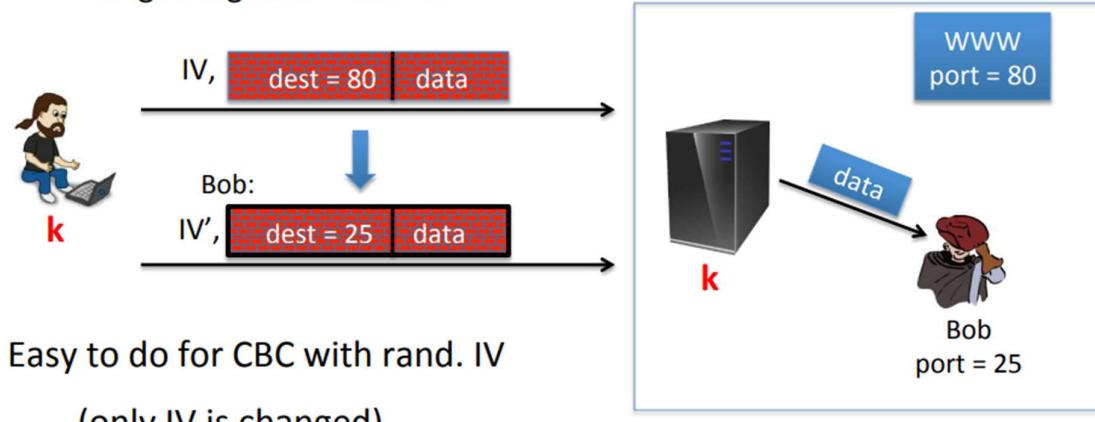
TCP/IP: (highly abstracted)



Khi bản tin gửi đi thay vì gửi dưới dạng bản rõ thì sẽ được đóng gói và các gói này sẽ được mã hóa trong các gói tin này sẽ chứa dữ liệu và cổng đích (*dest*) để gửi tới bên nhận. Khi hệ sử dụng tại tầng Transport có TCP/IP để giải mã kiểm tra xem bản tin được gửi tới đích là cổng nào và định tuyến gói tin data tới cổng đích, tuy nhiên trên đường truyền sẽ có các kẻ nghe lén sẽ sửa *cổng đích (dest)* mà bản tin gửi đi thành cổng của kẻ tấn công thay vì gửi tới port 80 thì hệ thống gửi tới port 25.

Note: attacker obtains decryption of any ciphertext

beginning with “dest=25”



Vậy thì cần tạo cơ chế để máy chủ TCP/IP nhận diện ra package gửi tới không còn toàn vẹn và cơ chế thay đổi *dest = 25* là các tấn công CCA – Chosen Ciphertext Attacks và tấn công CPA – Chosen Plaintext Attack

Ví dụ Sửa gói tin: Giả sử gói tin được mã hóa bằng CBC mode với IV ngẫu nhiên



Và ta cần sửa  $IV' = IV \oplus (\dots 80\dots) \oplus (\dots 25\dots)$  để  $m[0] = D(k, c[0]) \oplus IV = "dest=80\dots"$

*Kết luận chỉ ra rằng CPA bảo mật không đảm bảo bí mật khỏi phương pháp chủ động tấn công và chỉ nên dùng 2 chế độ: "Nếu thông điệp chỉ cần vẹn toàn và không cần bí mật thì sử dụng MAC và nếu thông điệp cần cả vẹn toàn và bí mật thì sử dụng mã có xác thực"*

## 7.2. Định nghĩa Mã xác thực

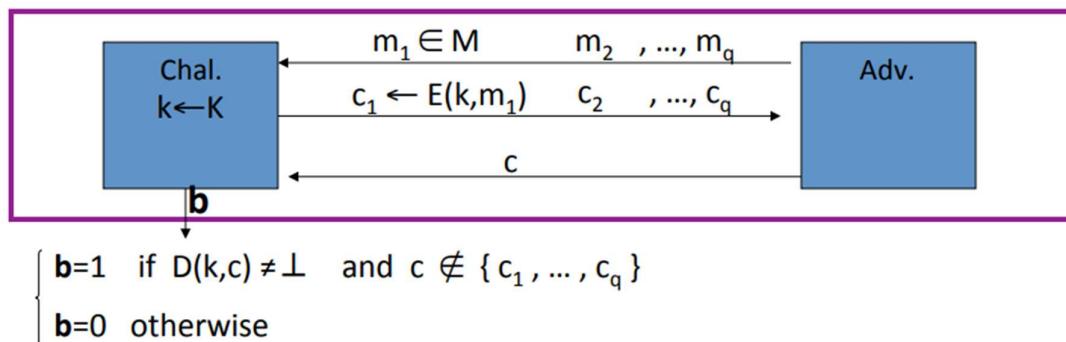
Mục tiêu của mã xác thực: Hệ thống mã hóa (E,D)

Thông thường:  $E: K \times M \times N \rightarrow C$

Nhưng:  $D: K \times C \times N \rightarrow M \cup \{\perp\}$  ( $\perp$  bản mã bị loại)

(*Nếu  $\perp$  phụ thuộc vào từng thuật toán AE – Authenticated Encryption khác nhau*)

Let  $(E, D)$  be a cipher with message space  $M$ .



Hình 62. Toàn vẹn Ciphertext

Kiểu tấn công CPA, kẻ tấn công cung cấp các  $m_i$  và nhận về các  $c_i$ . Dựa trên những ciphertext nhận được, kẻ tấn công sẽ soạn sẵn các ciphertext  $c$  thì yêu cầu đặt ra cho hệ thống kiểm định có khả năng kiểm tra thông qua  $\perp$ :

Nếu  $D(k, c) = \perp$  và  $\perp \neq \perp'$  tức là không nhận diện được tag không có quyền từ chối  $\rightarrow$  giải mã từ đó hệ thống không bảo mật. Để đảm bảo toàn vẹn thông điệp thì với mọi thuật toán hiệu quả A để lợi thế tấn công

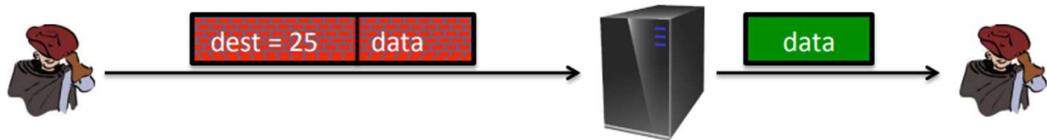
$Adv_{CI}[A, E] = Pr[Chal. Output = 1]$  là "không đáng kể"

*Kết luận một hệ mã  $[E, D]$  tạo ra authenticated encryption là có xác thực nếu an toàn dưới tấn công CPA và có tính toán vẹn thông điệp*

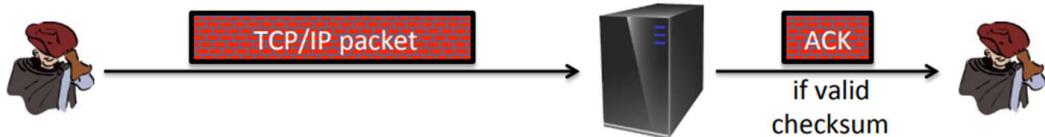
## 7.3. CCA

Tấn công bằng CCA gửi các bản mã  $c$  soạn trước và kẻ tấn công muốn giải mã các bản mã

- Often, adv. can fool server into decrypting **certain** ciphertexts (not c)

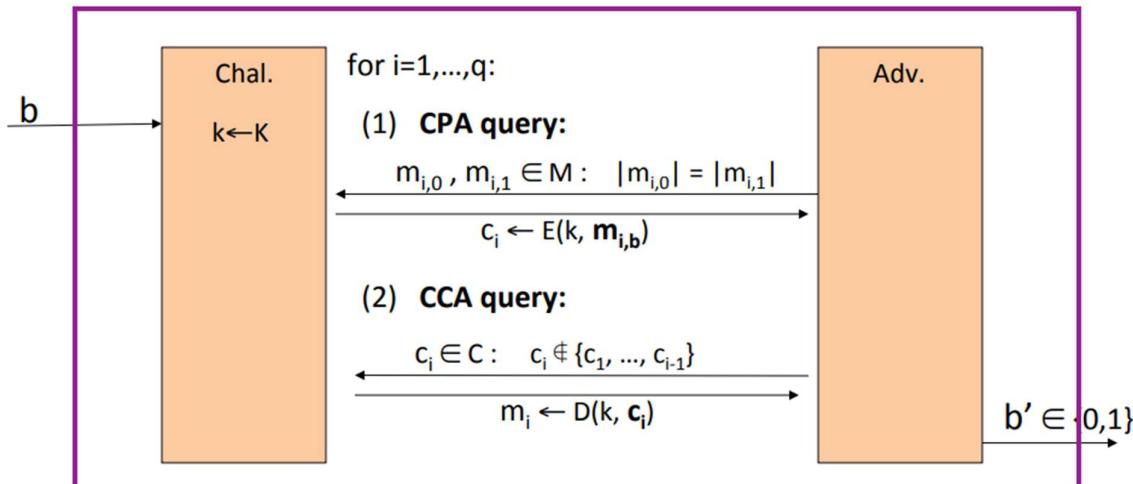


- Often, adversary can learn partial information about plaintext



Khả năng của kẻ tấn công bao gồm cả CPA và CCA đều có thể lấy được mã hóa của mọi thông điệp anh ta chọn, có thể giải mã mọi bản mã anh ta chọn c và mục đích của kẻ tấn công là phá an toàn ngữ nghĩa

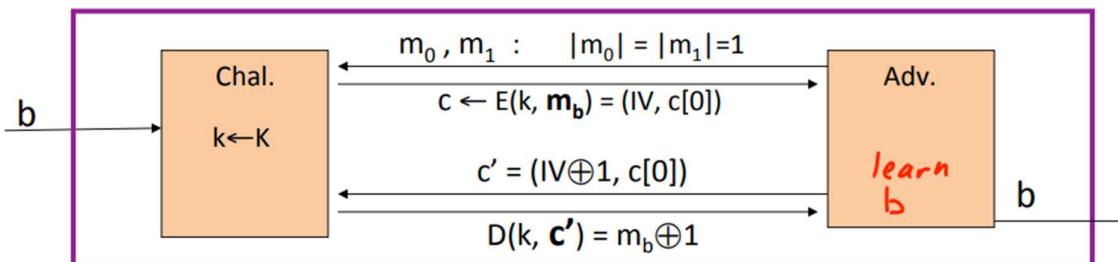
$E = (E, D)$  cipher defined over  $(K, M, C)$ . For  $b=0,1$  define  $\text{EXP}(b)$ :



Định nghĩa:  $E$  là CCA an toàn nếu với mọi thuật toán “hiệu quả”  $A$ :

$$\text{Adv}_{\text{CCA}}[A, E] = |\Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1]| \text{ “không đáng kể”}$$

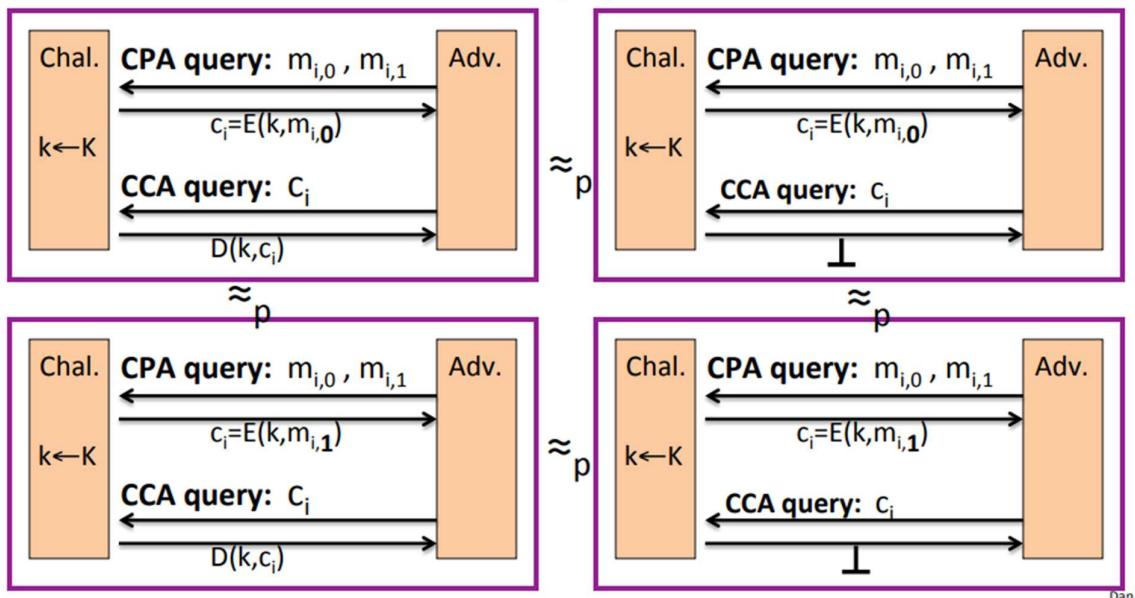
Ví dụ: CBC với IV ngẫu nhiên không an toàn trước tấn công CCA



Định lý: Xét  $(E, D)$  là hệ mã có xác thực. Vậy  $(E, D)$  là CCA an toàn. Đặc biệt với mọi thuật toán hiệu quả tấn công  $E$  dùng  $q$ -truy vấn, có tồn tại  $B_1, B_2$  thỏa mãn:

$$\text{Adv}_{\text{CCA}}[A, E] \leq 2q \cdot \text{Adv}_{\text{CI}}[B_1, E] + \text{Adv}_{\text{CPA}}[B_2, E]$$

Chứng minh:



### 7.3.1. Xây dựng từ hệ mã và MACs

Mã xác thực (AE) được giới thiệu vào năm 2000, API mật mã trước đó (ví dụ: MS-CAPI):

Cung cấp API mã hóa CPA-an toàn (Vd: CBC với IV ngẫu nhiên)

Cung cấp API cho MAC (Vd: HMAC)

- Kết hợp MAC và ENC (CCA)

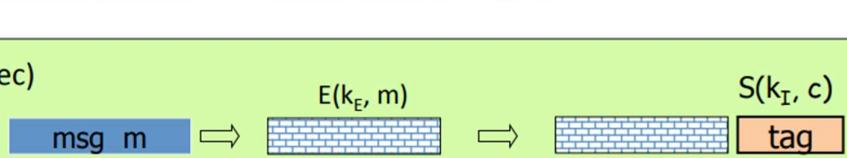
Encryption key  $k_E$ .    MAC key =  $k_I$

Option 1: (SSL)



Option 2: (IPsec)

**always  
correct**



Option 3: (SSH)



Hình 63. Kết hợp MAC và ENC (CCA)

Các cách truyền tin:

Với SSL: gán  $tag \leftarrow S(k_I, m)$  với  $m$  và mã hóa toàn bộ  $E(k_E, m || tag)$  (Encrypt and Mac)

Với Ipsec: mã hóa  $m$  với  $E(k_E, m)$  sau đó gửi  $E$  cùng với  $tag$  ( $E(k_E, m) || tag$ )(Encrypt then Mac)

Với SSH: tương tự như trên.

Nhận xét: Xét  $(E, D)$  là CPA an toàn và  $(S, V)$  là một MAC an toàn. Vậy thì:

Encrypt-then-MAC: luôn cho A.E

MAC-then-encrypt: có thể không an toàn chống tấn công CCA

tuy nhiên, khi  $(E, D)$  là rand-CTR mode hoặc rand-CBC vậy thì M-then-E cho phép A.E với rand-CTR mode, chỉ cần one-time MAC là đủ

➤ Các chuẩn (Ở mức cao)

- GCM
- CCM
- EAX

Các chuẩn này đều là chuẩn AEAD (*gán thêm dữ liệu đệm-tạo tính ngẫu nhiên đầu vào*)

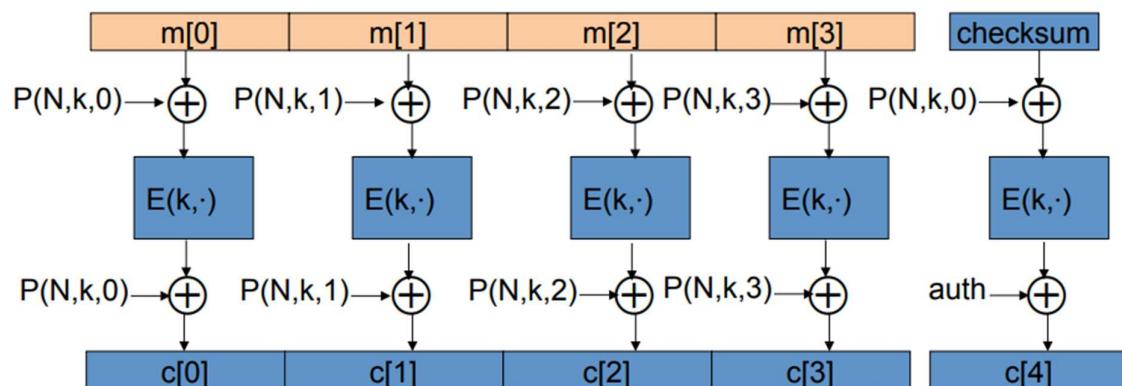
Ví dụ: API

```
int AES_GCM_Init(AES_GCM_CTX *ain,
                  unsigned char *nonce, unsigned long noncelen,
                  unsigned char *key, unsigned int klen )
```

```
int AES_GCM_EncryptUpdate(AES_GCM_CTX *a,
                           unsigned char *aad, unsigned long aadlen,
                           unsigned char *data, unsigned long datalen,
                           unsigned char *out, unsigned long *outlen)
```

### 7.3.2. OCB

Thuật toán AE hiệu quả theo cơ chế OCB



Hình 64. OCB

## CHƯƠNG 8. CÁC GIAO THỨC MẬT MÃ HỌC

*Giao thức mật mã* là nền tảng cho bảo mật thông tin. Chúng bao gồm các quy tắc và thủ tục sử dụng thuật toán mật mã để bảo mật thông tin liên lạc và bảo vệ dữ liệu. Giao thức mật mã mà hầu hết người dùng internet quen thuộc là giao thức Secure Sockets Layer (hay SSL), với hậu duệ của nó là Transport Layer Security (hay TLS) giao thức. Mục đích chính của giao thức mật mã bao gồm:

*Tính bảo mật:* Đảm bảo rằng chỉ những bên được ủy quyền mới có thể truy cập thông tin

*Tính toàn vẹn:* Đảm bảo thông tin không bị thay đổi trong quá trình truyền tải

*Xác thực:* Xác minh danh tính của các bên liên quan trong giao tiếp

*Không thể chối bỏ:* Ngăn chặn các bên phủ nhận sự tham gia của họ vào một giao dịch

Giao thức mật mã thường bao gồm một số thành phần chính:

*Thuật toán mật mã:* Các hàm toán học được sử dụng để mã hóa, giải mã và băm

*Khóa:* Các giá trị bí mật được sử dụng kết hợp với các thuật toán mật mã

*Các vector khởi tạo:* Các giá trị ngẫu nhiên được sử dụng để thêm tính ngẫu nhiên vào các quá trình mã hóa

*Chữ ký số:* Các lược đồ toán học được sử dụng để xác minh tính xác thực của tin nhắn hoặc tài liệu số

*Chứng chỉ:* Tài liệu kỹ thuật số liên kết khóa công khai với danh tính của một thực thể

Các giao thức mật mã hoạt động bằng cách kết hợp nhiều nguyên thủy và kỹ thuật mật mã khác nhau để đạt được mục tiêu bảo mật của chúng. Sau đây là tổng quan đơn giản về cách một giao thức điển hình có thể hoạt động:

*Khởi tạo:* Các bên liên quan thống nhất về giao thức và các thông số cần thiết.

*Trao đổi khóa:* Một phương pháp an toàn được sử dụng để trao đổi khóa mã hóa.

*Xác thực:* Danh tính của các bên đã được xác minh.

*Encryption:* Dữ liệu được mã hóa bằng các thuật toán và khóa đã thỏa thuận.

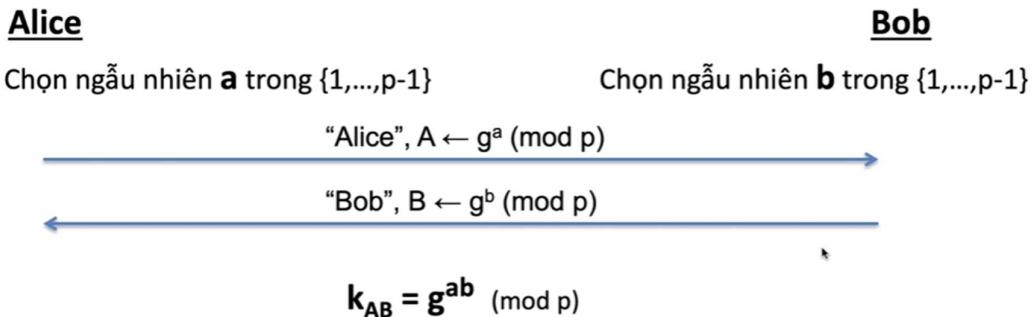
*truyền tải:* Dữ liệu được mã hóa được gửi qua mạng.

*Giải mã:* Người nhận giải mã dữ liệu bằng khóa của họ.

*Xác minh:* Tính toàn vẹn và xác thực của dữ liệu nhận được sẽ được kiểm tra

### 8.1. Giao thức Diffie-Hellman

Diffie-Hellman Key Exchange là một phương pháp được sử dụng để thiết lập một khóa bí mật chung giữa hai bên mà không cần phải truyền khóa đó qua mạng. Phương pháp này được phát triển bởi Whitfield Diffie và Martin Hellman vào năm 1976 và là một trong những nền tảng quan trọng của mật mã học hiện đại.



Hình 65. Thuật toán trao đổi khóa Diffie-Hellman

Hình 43 tóm tắt thuật toán trao đổi khóa Diffie–Hellman. Đối với phương pháp này, có hai số công khai được biết đến: một số nguyên tố  $p$  và một số nguyên  $g$  là căn nguyên nguyên thủy của  $p$ . Giả sử người dùng A và B muốn tạo ra một khóa bí mật giữa hai người.

Yêu cầu đầu vào:  $p$  là số nguyên tố lớn (vd 600 chữ số)

$g$  là một căn nguyên – nguyên thủy thuộc  $\{1, \dots, p\}$

$a$  là một số ngẫu nhiên thuộc  $\{1, \dots, p-1\}$

$b$  là một số ngẫu nhiên thuộc  $\{1, \dots, p-1\}$

Và giá trị  $a$  sẽ được giữ bí mật, không công khai, chỉ có mình Alice biết,  $b$  cũng tương tự chỉ có Bob biết. Cả hai cùng thực hiện phép Modulo  $A = g^a \text{ mod } p$ ,  $B = g^b \text{ mod } p$ , với  $A, B$  sẽ là 2 khóa công khai của Alice và Bob. Hai trao đổi khóa công khai này cho nhau, sau khi nhận khóa sẽ tiếp tục được modulo

$$K = g^{ab} \text{ mod } p$$

Tạo khóa bí mật giữa hai khóa riêng tư. Trong quá trình trao đổi công khai kẻ tấn công sẽ chỉ thấy các giá trị công khai  $p, g, A, B$  và lúc này nếu để tìm khóa bí mật riêng tư thì buộc phải tính *logarit ròng rạc*

$$b = d\log_{g,p}(B)$$

Kẻ tấn công sau đó có thể tính toán khóa K theo cách tương tự như cách Bob tính toán. Cụ thể, kẻ tấn công có thể tính toán K như sau

$$K = (B)^b \text{ mod } p$$

Trên thực tế để có thể tính toán, dự đoán đối với kẻ tấn công là rất khó thành công bởi do việc tính toán lũy thừa modulo một số nguyên tố là tương đối dễ dàng, thì việc tính toán logarit ròng rạc lại rất khó khăn. Đối với các số *nguyên tố lớn*, nhiệm vụ sau được coi là không khả thi.

Ví dụ: Cho các giá trị đầu vào như trong bảng và tính toán

Alice	Public Key	Bob
$a = 97$	$p = 353, g = 3$	$b = 233$
$A = g^a \text{ mod } p$ $= 3^{97} \text{ mod } 353 = 40$	$p, g$ $A, B$	$B = g^b \text{ mod } p$ $= 3^{233} \text{ mod } 353 = 248$

## IT3150 – Project 1

$K = (g)^{ab} \bmod p$ $= 248^{97} \bmod 353 = 160$	$p, g$ $A, B$	$K = (g)^{ab} \bmod p$ $= 40^{233} \bmod 353 = 160$
Khóa bí mật chung: $K = 160$		

## CHƯƠNG 9. MẬT MÃ KHÓA CÔNG KHAI

Một mật mã khóa công khai bao gồm bộ ba thuật toán (G, E, D)

- G(K): Thuật toán *ngẫu nhiên* có output cặp khóa ( $pk, sk$ )
- E( $pk, m$ ): Thuật toán *ngẫu nhiên* nhận  $m \in M$  và output  $c \in C$
- D( $sk, c$ ): Thuật toán *đơn định* nhận  $c \in C$  và outputs  $m \in M$  hoặc ⊥

Tính đúng đắn:  $\forall(pk, sk)$  sinh bởi G:  $\forall m \in M: D(sk, E(pk, m)) = m$

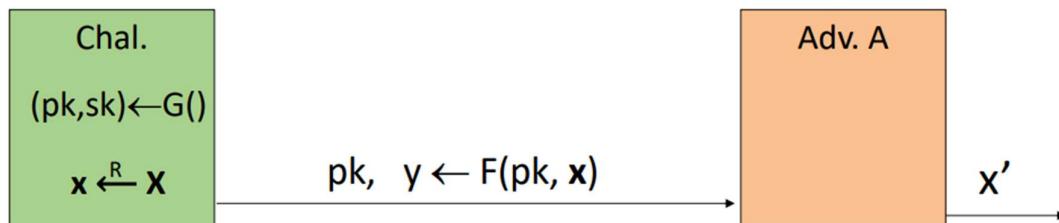
### 9.1. TDF

Định nghĩa: Hàm cửa sập  $X \rightarrow Y$  là bộ ba thuật toán hiệu quả (G, F,  $F^{-1}$ )

- G(K): Thuật toán *ngẫu nhiên* có output cặp khóa ( $pk, sk$ )
- F( $pk, \cdot$ ): Thuật toán *đơn định* định nghĩa một hàm  $X \rightarrow Y$
- $F^{-1}(sk, \cdot)$ : Nghịch đảo của F( $pk, \cdot$ ):  $Y \rightarrow X$

Nghĩa:  $\forall(pk, sk)$  sinh bởi G thì  $\forall x \in X: F^{-1}(sk, F(pk, x)) = x$

Một TDF an toàn:  $(G, F, F^{-1})$  là an toàn nếu F( $pk, \cdot$ ) là hàm “một chiều” (*one-way function*) có thể tính theo chiều xuôi nhưng không thể tính nghịch đảo mà không có  $sk$ . Ví dụ:



Kẻ tấn công gửi nhận được  $pk$  và  $y$ . Để cửa sập này an toàn nếu với mọi thuật toán hiệu quả A ta có lợi thế tấn công:

$\text{Advow}[A, F] = \Pr[x = x'] = \text{"không đáng kể"}$

Xây dựng hệ mật khẩu công khai (*public key*) từ TDFs yêu cầu:

- $(G, F, F^{-1})$ : TDF-secure  $X \rightarrow Y$
- $(E_s, D_s)$ : Hệ mật mã đối xứng an toàn trên  $(K, M, C)$
- H:  $X \rightarrow K$ : Hàm băm

Ta xây dựng hệ mật khẩu công khai (G, E, D)

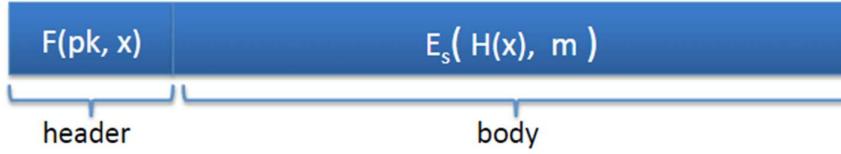
$E(pk, m)$	$D(sk, (y, c))$
$x \leftarrow X, y \leftarrow F(pk, x)$ $k \leftarrow H(x), c \leftarrow E_s(k, m)$	$x \leftarrow F^{-1}(sk, y),$ $k \leftarrow H(x), m \leftarrow D_s(k, c)$
Output: $(y, c)$	Output: $m$

Lưu ý: *Cách sử dụng sai TDF – Không được mã hóa bằng cách áp dụng F để mã hóa bản rõ:*

<u>E( pk, m ) :</u>	<u>D( sk, c ) :</u>
output $c \leftarrow F(pk, m)$	output $F^{-1}(sk, c)$

Bởi: đây là một hệ mã *đơn định* nên có nhiều cách tấn công

In pictures:



➔ *Định lý bảo mật:* Nếu  $(G, F, F^{-1})$  là một TDF an toàn,  $(E_S, D_S)$  cung cấp mã hóa xác thực và  $H: X \rightarrow K$  là một “oracle ngẫu nhiên” thì  $(G, E, D)$  là an toàn CCA

## 9.2. RSA

Định nghĩa:

$(E_S, D_S)$ : Hệ mật mã đối xứng an toàn trên  $(K, M, C)$

$H: Z_N \rightarrow K$  với  $K$  là không gian khóa của  $(E_S, D_S)$

Bộ ba thuật toán:

<b>G()</b>	Sinh tham số RSA: $pk = (N, e)$ , $sk = (N, d)$
<b>E(pk,m)</b>	Chọn số ngẫu nhiên $x \in Z_N$ $Y \leftarrow RSA(x) = x^e$ , $k \leftarrow H(x)$ Output: $(y, Es(k, m))$
<b>D(sk, (y,c))</b>	Output: $D_S(H(RSA^{-1}(y)), c)$

## 9.3. RSA-TDP

$G()$ : chọn ngẫu nhiên các số nguyên tố  $p, q \sim 1024$  bits. Thiết lập  $N=p*q$ , chọn 2 số nguyên dương  $e, d$  thỏa mãn  $e*d = 1 \pmod{\phi(N)}$

$$\phi(N) = (p-1)(q-1)$$

Output:  $pk = (N, e)$ ,  $sk = (N, d)$

$$\Rightarrow F(pk, x): Z_N^* \rightarrow Z_N^* ; \quad RSA(x) = x^e \text{ (in } Z_N\text{)}$$

$$\Rightarrow F^{-1}(sk, y): y^d; \quad y^d = RSA(x)^d = x^{ed} = x^{k\phi(N)+1} = (x^{\phi(N)})^k * x = x$$

Giả sử RSA là hoán vị “một chiều” thì với mọi thuật toán hiệu quả A:

$$Pr[ A(N, e, y) = y ] < "không đáng kể"$$

ở đó  $p, q$  là số nguyên tố  $\sim 1024$  bits,  $N \leftarrow pq$ ,  $y \leftarrow Z_N^*$

## 9.4. One-way function

Một hàm f:  $X \rightarrow Y$  là một chiều nếu tồn tại một thuật toán hiệu quả để đánh giá  $f(\bullet)$ , nhưng việc đảo ngược f thì rất khó: Với mọi thuật toán hiệu quả A và  $x \leftarrow X$  thì:

$$Pr[F(A(f(x))) = f(x)] < "không đáng kể"$$

Các hàm không phải 1 chiều:  $f(x) = x$ ,  $f(x) = 0$

Xét ví dụ hàm một chiều RSA:

- Chọn các số nguyên tố lớn  $p, q \sim 1024$  bits và  $N = p * q$
- Chọn các số nguyên dương  $e, d$  sao cho  $e * d = 1 \pmod{\phi(N)}$

Định nghĩa:  $f: Z_N^* \rightarrow Z_N^*$  và  $f(x) = x^e$  trong  $Z_N$

Định lý:  $f$  là hàm một chiều theo giả thuyết RSA

Tính chất phân phối:  $f(x \cdot y) = f(x) \cdot f(y)$  và hàm  $f$  có một khóa bí mật (trapdoor) cho phép đảo ngược hàm khi biết khóa riêng  $d$ .

## 9.5. Hệ mật Elgamal (*hiện đại*)

Hệ mật Elgamal (*theo gốc nghìn mói*) bao gồm

- $G$ : nhóm vòng cấp  $N$
- $(E_S, D_S)$ : mã đối xứng an toàn trên  $(K, M, C)$
- $H: G^2 \rightarrow K$ : Hàm băm

Lúc này, ta xây dựng một hệ mật khóa công khai (Gen, E, D) trong đó sinh khóa Gen được chọn ngẫu nhiên phần tử sinh  $g$  trong  $G$ , chọn một số ngẫu nhiên  $a \in Z_N$  và kết quả đầu ra  $sk = a$ ,  $pk = (g, h = g^a)$

$E(pk=(g,h), m)$	$D(sk = a, (u,c))$
$b \leftarrow Z_n, u \leftarrow g^b, v \leftarrow h^b$	$v \leftarrow u^a$
$k \leftarrow H(u,v), c \leftarrow E_S(k,m)$	$k \leftarrow H(u,v), m \leftarrow D_S(k,c)$
Output: $(u,c)$	Output: $m$

Dánh giá hiệu năng hệ mật:

**E( pk=(g,h), m ) :**  
 $b \leftarrow Z_n, u \leftarrow g^b, v \leftarrow h^b$

**D( sk=a, (u,c) ) :**  
 $v \leftarrow u^a$

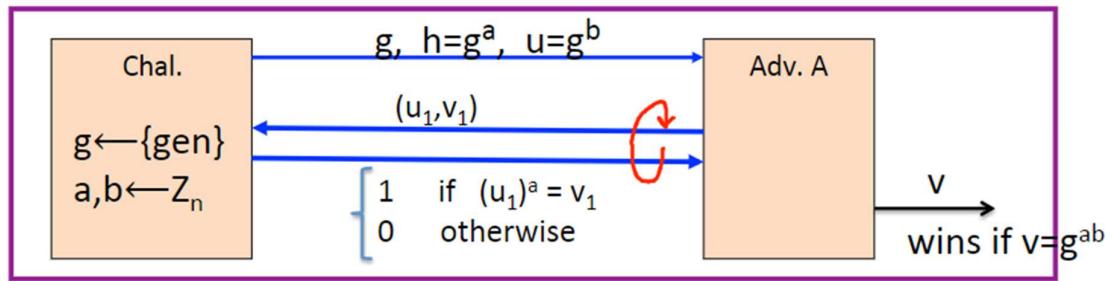
**Mã hóa:** 2 phép lấy mũ. (cơ sở cố định)

- Có thể tính trước  $[g^{(2^i)}, h^{(2^i)} \text{ for } i=1, \dots, \log_2 n]$
- Tốc độ nhanh gấp 3x (hoặc hơn)

**Decryption:** 1 phép lấy mũ. (cơ sở thay đổi)

*Hình 66. Hiệu năng Elgamal*

➤ Elgamal – CCA



Hình 67. Elgamal – CCA

Để chứng minh tính bảo mật trong trường hợp tấn công bằng CCA thì cần một giả thuyết mạnh hơn, Giả thuyết Interactive Diffie-Hellman (IDH) trong nhóm G. Giả thuyết này được coi là đúng nếu với voi thuật toán A hiệu quả

$$Pr[A \text{ output } g^{ab}] < \text{"không đáng kể"}$$

Nếu IDH đúng, điều này cho thấy rằng việc tìm ra  $g^{ab}$  từ  $g, g^a, g^b$  là rất khó khăn, ngay cả đối với các thuật toán hiệu quả.

- ➔ Định lý bảo mật: nếu giả thuyết IDM đúng thì trong nhóm  $(G, E, D)$  cung cấp mã hóa xác thực và H:  $G^2 \rightarrow K$  là một “ oracle ngẫu nhiên”, thì ElGamal là an toàn trong chế độ CCA

## CHƯƠNG 10. KẾT LUẬN

Mật mã học là một lĩnh vực quan trọng trong bảo mật thông tin, đóng vai trò thiết yếu trong việc bảo vệ dữ liệu và đảm bảo an toàn trong giao tiếp. Qua việc sử dụng các thuật toán mã hóa, mật mã học không chỉ giúp bảo vệ thông tin khỏi việc truy cập trái phép mà còn xác thực danh tính của các bên tham gia giao dịch. Các ứng dụng của mật mã học rất đa dạng, từ việc bảo mật thông tin trong giao tiếp điện tử đến việc bảo vệ dữ liệu trên các nền tảng lưu trữ và trong các giao thức bảo mật như SSL/TLS. Những thuật toán mã hóa hiện đại như AES và RSA đã chứng minh được tính hiệu quả và độ an toàn cao, mặc dù vẫn cần phải liên tục cải tiến để đối phó với những mối đe dọa mới. Mặt khác, mật mã học cũng không phải là một giải pháp hoàn hảo. Độ an toàn của nó phụ thuộc vào việc triển khai đúng cách và bảo mật khóa. Do đó, việc nâng cao nhận thức về mật mã học và áp dụng các biện pháp bảo mật phù hợp là rất cần thiết trong thời đại công nghệ số ngày nay.

## **TÀI LIỆU THAM KHẢO**

- <https://www.coursera.org/learn/crypto>
- <https://cs251.stanford.edu/>
- <https://github.com/smartcontractkit/full-blockchain-solidity-course-js>
- cryptography-and-network-security-principles-and-practice-7th-global-edition